Université de Paris VII - Denis Diderot UFR d'informatique

Construction et Analyse de Schémas Cryptographiques

THÈSE d'HABILITATION

présentée et soutenue publiquement le 30 mars 2006

pour l'obtention du

Diplôme d'Habilitation à Diriger des Recherches

par

Jean-Sébastien Coron

Composition du jury :

Rapporteurs :	Dan Boneh Anca Muscholl Moti Yung
Examinateurs :	David Naccache David Pointcheval
Directeur de recherche :	Jacques Stern

à Florence, Inès et Louis

Remerciements

Je remercie tout d'abord Jacques Stern. Il avait déjà dirigé ma thèse de doctorat, et je suis très heureux qu'il ait bien voulu diriger aussi cette thèse d'habilitation.

La plupart des travaux présentés dans ce mémoire ont été réalisés lorsque j'étais salarié de Gemplus. Je remercie donc tout particulièrement David Naccache qui m'a accueilli dans son équipe en 1998; il est à l'origine de plusieurs travaux présentés dans ce mémoire. Je remercie aussi mes anciens collègues Pascal, Helena, Alexei, Nora et Florence, avec qui j'ai eu beaucoup de plaisir à travailler. Je remercie Marc Joye pour son aide dans la composition de ce document.

Je remercie également les membres du groupe de recherche en Cryptographie de l'ENS, notamment Phong Nguyen et David Pointcheval, avec qui j'ai eu d'intéressantes discussions.

Je remercie Dan Boneh, Anca Muscholl et Moti Yung pour avoir accepté d'être rapporteurs, ainsi que David Naccache et David Pointcheval qui ont bien voulu être membres du jury.

Enfin, je remercie tous mes co-auteurs de publications: Claude Barral, Eric Brier, Julien Cathalo, Christophe Clavier, Nora Dabbous, Yevgeniy Dodis, Louis Goubin, Helena Handschuh, Marc Joye, Paul Kocher, François Koeune, David Lefranc, Alexander May, David M'Raïhi, Cécile Malinaud, David Naccache, Pascal Paillier, David Pointcheval, Guillaume Poupard, Prashant Puniya, Julien Stern, Alexei Tchulkine et Christophe Tymen.

Avant-Propos

Ce document constitue le dossier en vue de l'obtention de l'habilitation à diriger des recherches soumis à l'université Paris 7 - Denis Diderot.

Les travaux présentés dans cette thèse portent sur les deux thèmes principaux suivants : 1) la formalisation de la sécurité et la construction de schémas cryptographiques à sécurité garantie, et 2) la cryptanalyse mathématique de schémas cryptographiques.

Le document est composé de trois parties :

1. Une introduction aux notions de base pour ces deux thèmes principaux et une présentation de mes travaux dans ces deux thèmes.

2. Un curriculum vitae et une liste complète de mes publications.

3. Une annexe regroupant mes principaux articles publiés depuis ma thèse.

Table des matières

Partie I

Construction et Analyse de Schémas Cryptographiques

Introduction à la cryptologie

1 Aperçu général de la cryptographie

1.1 Aperçu historique

La cryptologie a très longtemps été considérée comme une arme de guerre. C'est au cours de ces trente dernières années qu'elle est devenue un thème de recherche scientifique. La cryptologie est une discipline liée à beaucoup d'autres, comme la théorie des nombres, la théorie de la complexité, la théorie de l'information, ou encore les codes correcteurs d'erreurs. Les applications principales de la cryptologie sont bien sûr le chiffrement des messages pour en garantir la confidentialité, mais aussi l'authentification à distance, la signature de documents numériques, et le contrôle de l'intégrité des données transmises sur un réseau informatique.

Il existe deux modèles distincts pour le chiffrement de l'information. Le premier est le chiffrement symétrique, où la même clé secrète est partagée entre deux correspondants. La même clé permet à la fois de chiffrer et de déchiffrer le message. L'algorithme de chiffrement symétrique le plus connu est l'algorithme DES (Data Encryption Standard) [1], adopté en 1976. La taille de la clé étant cependant limitée à 56 bits, l'algorithme DES est aujourd'hui vulnérable aux attaques par recherche exhaustive. Par conséquent, le bureau des standards Américain a lancé en 1997 un appel à proposition pour définir un remplaçant du DES dénommé AES (Advances Encryption Standard) [63], et utilisant une clé allant de 128 à 256 bits. C'est l'algorithme Rijndael [33] qui a finalement été sélectionné au mois d'octobre 2000; cet algorithme est appelé à remplacer le DES dans la plupart des applications. Nous référons le lecteur aux nombreux ouvrages généraux sur la cryptographie ([71], [56] et [74]) qui offrent un tour d'horizon relativement complet des techniques de chiffrement symétrique.

Le deuxième modèle est le chiffrement à clé publique ou chiffrement asymétrique, un concept inventé en 1976 par Whitfield Diffie et Martin Hellman [36] et qui a révolutionné la cryptologie. Dans ce modèle, chaque utilisateur possède deux clés: une clé publique permettant le chiffrement, et une clé privée permettant le déchiffrement. Pour chiffrer un message, on utilise donc la clé publique du destinataire, qui est le seul à pouvoir ensuite déchiffrer le message avec la clé privée correspondante. La cryptographie à clé publique résout ainsi le problème de la distribution des clés: deux personnes peuvent communiquer de façon confidentielle sans jamais s'être rencontrées auparavant.

Le premier exemple de schéma de chiffrement à clé publique a été l'algorithme RSA [69], proposé en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman. C'est actuellement l'algorithme de chiffrement à clé publique le plus utilisé dans le monde. La sécurité de RSA repose essentiellement sur la difficulté de factoriser des grands nombres. En fait, on ne connaît pas de preuve que cette factorisation est difficile à obtenir, mais aucune méthode efficace n'est apparue pour l'instant. Nous décrirons cet algorithme plus en détail au paragraphe 2.

1.2 Objectifs de la cryptographie

Le but de la cryptographie est de construire des schémas qui accomplissent un certain objectif malgré la présence d'un attaquant. Les objectifs que l'on cherche à atteindre sont généralement la confidentialité, l'intégrité et l'authenticité.

Confidentialité. La confidentialité permet de protéger le contenu des informations sauvegardées ou transmises sur un réseau. Seules les personnes autorisées doivent pouvoir accéder aux informations ainsi protégées. Le *chiffrement de l'information* permet de résoudre le problème de la confidentialité: une personne souhaitant transmettre un message lui applique au préalable une fonction dite de chiffrement, et transmet le résultat au destinataire. Ce dernier retrouve le message original en utilisant une fonction de déchiffrement.

Dans le modèle de la cryptographie à clé secrète (aussi appelé chiffrement symétrique), les deux parties partagent la même clé de chiffrement et de déchiffrement, qui doit être gardée secrète. Les deux personnes jouent ainsi un rôle symétrique.

Dans le modèle de la cryptographie à clé publique, le chiffrement est publique tandis que le déchiffrement est confidentiel. Pour envoyer un message chiffré, l'auteur du message applique une fonction de chiffrement qui utilise la clé publique du destinataire. Seul le destinataire peut retrouver le message original à l'aide de la clé privée correspondant à sa clé publique. Les deux clés sont liées mathématiquement, mais il doit être impossible dans la pratique de retrouver la clé privée à partir de la clé publique.

Les algorithmes à clé secrète et à clé publique ont chacun leur domaine d'utilisation avec leurs avantages et inconvénients. Les algorithmes à clé secrète sont en général beaucoup plus rapides, et ont des clés plus courtes. L'avantage essentiel des algorithmes à clé publique réside dans la gestion des clés: seule la clé privée est maintenue secrète, les clés ont une durée de vie plus longue, et le nombre total de clés requises pour que des personnes puissent communiquer entre elles sur un réseau est considérablement réduit.

Dans la pratique, on peut combiner les avantages de la clé publique et de la clé secrète. Par exemple, le chiffrement à clé publique permet d'établir une clé secrète entre deux entités A et B qui peuvent ensuite communiquer de façon sécurisée avec cette clé. Dans ce scénario, l'entité A choisit aléatoirement une clé secrète, dite clé de session, et la chiffre en utilisant la clé publique de B. Ce dernier retrouve la clé de session en utilisant sa clé privée, et ensuite A et B peuvent communiquer de façon sécurisée en utilisant la clé de session en chiffrement symétrique. Des applications très populaires telles que SSL [47] mettent en oeuvre ce principe plusieurs millions de fois par jour.

Intégrité. Le problème de l'intégrité est le contrôle du contenu : on veut pouvoir détecter toute modification, accidentelle ou intentionnelle, des données sauvegardées ou transmises. On résout ce problème à l'aide d'une *fonction de hachage*, qui à une donnée de longueur arbitraire associe une donnée de taille fixe appelée *empreinte* ou *haché*. Ainsi, le problème du maintien de l'intégrité d'une donnée de taille arbitraire se réduit à celui d'une donnée de taille fixe, ce qui est beaucoup plus facile à réaliser en pratique. Il faut pour cela que la fonction de hachage soit *résistante aux collisions*, c'est à dire qu'il doit être impossible en pratique de trouver deux messages distincts qui donnent un haché de même valeur; personne ne peut alors modifier le message en conservant la même empreinte. Généralement,

on demande aussi à une fonction de hachage d'être à *sens unique*, c'est à dire qu'il doit être impossible étant donné un haché de trouver un message qui donne ce haché.

Authenticité. La notion d'authenticité s'applique à la fois aux personnes, on parle dans ce cas d'identification, et aux documents, ce qui correspond à l'authentification.

Pour s'identifier, un individu prouve qu'il connaît une information secrète. Une notion particulièrement élégante est celle de preuve interactive à divulgation nulle de connaissance (preuve zero-knowledge), dans laquelle un individu (le prouveur) convainc une autorité (le vérifieur) qu'il possède une information secrète, sans révéler aucune information sur cette donnée secrète. Ainsi, il n'est pas possible qu'un espion écoutant la ligne se fasse ensuite passer pour le prouveur, car l'espion n'apprend rien au cours de l'échange.

L'authentification d'un document permet de prouver son caractère authentique, c'est à dire son lien avec une entité précise. On garantit ainsi l'origine de l'information contenue dans le document. Les techniques utilisées sont les codes d'authentification de message (notés de façon usuelle MAC pour *Message Authentification Code*) en clé secrète, et la signature électronique en clé publique. Ces techniques assurent implicitement l'intégrité du document.

Dans un code d'authentification de message, l'entité à l'origine du document partage la même clé secrète avec l'entité qui en vérifie l'origine. La même clé secrète permet donc de prouver l'authenticité du document et d'en vérifier l'intégrité.

La signature numérique d'un document électronique permet de lier le contenu du document à son signataire, en assurant à la fois l'intégrité du document (le document n'a pas été modifié) et l'authenticité du signataire (c'est bien lui qui a signé). La vérification de la signature est publique: n'importe qui peut la vérifier en utilisant la clé publique du signataire. On obtient ainsi la propriété de non-répudiation de la signature: le signataire ne peut pas nier par la suite avoir signé le document. Le lecteur peut se référer à [56] pour une étude plus précise des codes d'authentification de message et des algorithmes de signature électronique.

2 Le schéma RSA

2.1 Chiffrement et signature RSA

Le cryptosystème RSA a été inventé en 1977 par Ron Rivest, Adi Shamir et Leonard Adleman [69]. Premier algorithme à clé publique découvert, c'est actuellement le plus utilisé dans le monde. L'algorithme RSA se retrouve dans de nombreux produits commerciaux liés à la confidentialité des transactions sur Internet, à la confidentialité et l'authenticité du courrier électronique, ou à la connection sécurisée avec un ordinateur distant.

Depuis sa parution il y a 28 ans, le cryptosystème RSA a fait l'objet de nombreuses études mettant en évidence certaines vulnérabilités, mais aucune n'a permis un cassage total. Les attaques mises en oeuvre ont surtout montré le danger d'une utilisation incorrecte de RSA. Dans ce chapitre, nous commençons par rappeler la définition de RSA et nous décrivons ensuite les principales attaques connues. Certaines attaques plus évoluées, basées sur le théorème de Coppersmith, seront présentées au chapitre 4.

Un *module* RSA est le produit $N = p \cdot q$ de deux grands nombres premiers p et q de taille voisine. La taille typique d'un module RSA est de 1024 bits, chaque facteur premier ayant une taille de 512 bits. Soient e et d deux entiers tels que $e \cdot d = 1 \bmod \phi(N)$, où $\phi(N) = (p-1) \cdot (q-1)$ est la fonction d'Euler. Rappelons que $\phi(N)$ est l'ordre du groupe multiplicatif $\mathbb{Z}_N^*$. On appelle e l'*exposant de chiffrement* et d l'*exposant de déchiffrement*. La paire (N, e) est la clé publique, qui permet à chacun de chiffrer un message, et la paire (N, d) est la clé privée, permettant à celui qui la possède de déchiffrer un message chiffré.

Un *message* est un entier $m \in \mathbb{Z}_N^*$. Pour chiffrer le message m, on calcule

$$c = m^e \bmod N \tag{1}$$

Ainsi, comme le module N et l'exposant e sont publiques, tout le monde peut chiffrer un message. Pour déchiffrer c, on calcule:

$$m = c^d \bmod N$$

Seule la personne possédant l'exposant privé de déchiffrement peut donc déchiffrer le message. On retrouve effectivement le message m car:

$$c^d = m^{e \cdot d} = m^{1+k \cdot \phi(N)} = m \bmod N$$

Dans la pratique, pour obtenir de meilleures performances, le déchiffrement RSA est réalisé en utilisant le théorème du reste Chinois (CRT). Pour calculer $m = c^d \bmod N$, on calcule séparément le message m modulo p et modulo q, en calculant $m_p = c^{d_p} \bmod p$ et $m_q = c^{d_q} \bmod q$, avec $d_p = d \bmod p-1$ et $d_q = d \bmod q-1$. En utilisant le théorème du reste Chinois, on obtient l'unique message $m \in \mathbb{Z}_N$ tel que $m = m_p \bmod p$ et $m = m_q \bmod q$. Une exponentiation modulaire ayant une complexité cubique en la taille du module, chacune des deux exponentiations est 8 fois plus rapide que l'exponentiation $m = c^d \bmod N$. Il en résulte donc un gain d'un facteur 4 en temps de calcul.

En réalité, nous verrons par la suite qu'avant de chiffrer un message avec RSA, il faut obligatoirement lui appliquer un encodage particulier, destiné à prévenir la fuite d'information sur ce message. Ce problème sera abordé en détail aux chapitres 2 et 3 lorsque nous expliquerons comment se formalise la sécurité d'un schéma de chiffrement à clé publique.

Le cryptosystème RSA permet aussi de générer des signatures électroniques. Pour signer un message $m \in \mathbb{Z}_N^*$, le signataire utilise sa clé privée (N, d) pour obtenir la signature:

$$s = m^d \bmod N$$

Seul le signataire possédant la clé privée (N, d) est donc capable de signer le message m. On vérifie la validité de la signature s à l'aide de la clé publique (N, e), en s'assurant que:

$$m = s^e \bmod N$$

Ainsi, n'importe qui peut vérifier la signature s de m. En réalité, comme pour le chiffrement, il faut aussi appliquer au message m à signer un encodage particulier. Nous verrons au chapitre 2 comment se formalise la sécurité d'un schéma de signature, et au chapitre 3 les principales constructions.

Pour obtenir une paire de clés RSA, on génère aléatoirement deux nombres premiers p et q de taille $n/2$ bits et on les multiplie pour obtenir le module $N = p \cdot q$. Ensuite, étant donné un exposant de chiffrement $e < \phi(N)$, on calcule $d = e^{-1} \bmod \phi(N)$ en utilisant l'algorithme d'Euclide étendu. On génère habituellement un nombre premier aléatoire p de taille $n/2$-bits en générant aléatoirement des entiers de $n/2$-bits et en testant leur primalité à l'aide d'un test de primalité probabiliste [56].

On montre facilement que la connaissance de la clé privée de RSA est équivalente au problème de la factorisation. En effet, il existe un algorithme probabiliste dû à Miller [57] qui étant donné (N, e, d) renvoie la factorisation de N en temps polynomial. Nous verrons au chapitre 4 que cette équivalence peut être rendue *déterministe* en utilisant une variante du théorème de Coppersmith.

L'algorithme de chiffrement RSA conduit à définir la fonction de chiffrement RSA:

$$f : \mathbb{Z}_N^* \to \mathbb{Z}_N^*, x \to x^e \bmod N$$

Si l'entier d est connu la fonction peut être facilement inversée:

$$f^{-1}(y) = y^d \bmod N$$

Le problème de la sécurité de RSA peut alors se formuler de la façon suivante :

Problème RSA: Etant donné (N, e) et un entier $y \in \mathbb{Z}_N$, trouver x tel que $x^e = y \bmod N$.

Un problème ouvert important consiste à savoir si le problème RSA est aussi difficile que le problème de la factorisation. Il existe des arguments qui indiquent le contraire [12], pour un exposant e petit. Cependant, le problème RSA peut fort bien être un problème difficile sans pour autant être équivalent à la factorisation.

2.2 Attaques contre RSA

Factorisation. La première attaque possible contre le cryptosystème RSA consiste à tenter de factoriser le module N pour retrouver les facteurs premiers p et q. L'algorithme de factorisation le plus rapide actuellement est la méthode du *crible algébrique* [51], dont le temps de calcul heuristique pour factoriser un module RSA N est donné par:

$$\exp\left((1.923 + \circ(1)) \ln(N)^{1/3} \ln(\ln N)^{2/3}\right)$$

Le record actuel est la factorisation d'un module RSA de 663 bits (RSA-200) obtenu en mai 2005. Pour des applications devant rester sûres pendant de nombreuses années, on recommande l'utilisation de modules RSA de taille au moins 2048 bits.

Cependant, il existe d'autres attaques possibles contre le cryptosystème RSA qui ne nécessitent pas de s'attaquer à la factorisation du module. Ces attaques ne mettent pas en danger le cryptosystème RSA lui-même mais illustrent plutôt le danger d'une utilisation incorrecte du schéma. On pourra se reporter à [9] pour une synthèse de ces attaques.

Forge existentielle. Etant donné un module N, si aucun encodage particulier n'est appliqué au message m, la signature de m est donnée par

$$s = m^d \bmod N$$

On voit donc que pour tout entier $r \in \mathbb{Z}_N$, la signature du message $m = r^e \bmod N$ est égale à r. Il est donc possible de produire une signature valide d'un message sans connaître l'exposant privé d : c'est ce que l'on appelle une forge existentielle. Dans la pratique, pour éviter cette attaque, on applique au préalable un encodage particulier au message m pour qu'il soit difficile de trouver un entier r tel que $r^e \bmod N$ respecte ce encodage. Nous verrons par la suite des exemples d'encodage possible.

Attaque par masquage. Soit (N, d) la clé privée d'Alice correspondant à la clé publique (N, e). Supposons qu'un attaquant dénommé Marvin désire faire signer le message m à Alice. En lisant le message m, Alice refuse de le signer. Marvin génère alors un entier aléatoire $r \in \mathbb{Z}_N$ et calcule $m' = m \cdot r^e \bmod N$. Il demande ensuite à Alice de signer le message aléatoire m'. Si Alice accepte, Marvin obtient:

$$s' = m'^d \bmod N = r \cdot m^d \bmod N$$

et peut donc calculer la signature $s = m^d = s'/r \bmod N$ du message m.

Il est facile de transposer l'attaque précédente dans le cas du chiffrement RSA. Un attaquant voulant obtenir le déchiffrement $m = c^d \bmod N$ d'un chiffré c peut choisir un entier aléatoire r et demander le déchiffrement de $c' = r^e \cdot c \bmod N$. Avec la réponse $m' = (c')^d \bmod N$, il obtient facilement le message original $m = m' \cdot r^{-1} \bmod N$. Pour palier cette attaque, on ajoute de la redondance au message m à chiffrer, pour qu'il soit difficile à un attaquant de trouver un entier r tel que le chiffré c' corresponde à un clair m' qui respecte cette redondance.

L'attaque précédente peut s'étendre au cas où l'attaquant n'obtient que le bit de poids faible du message clair [44]. Il existe en effet un algorithme permettant de déchiffrer tout message, en utilisant un oracle qui donne le bit le moins significatif du message clair correspondant au chiffré.

Pour la signature, deux méthodes peuvent permettre de se prémunir contre de telles attaques. La première méthode consiste à ajouter de la redondance au message, par exemple une chaîne de caractère fixe. Nous verrons au paragraphe suivant diverses attaques contre cette méthode. La deuxième méthode consiste à hacher le message avant de le signer. Nous verrons que si la taille du haché est trop courte, il est malgré tout possible de forger des signatures.

Attaques sur les signatures RSA avec redondance. Dans ce paragraphe, on considère les signatures RSA obtenues de la manière suivante :

$$s = R(m)^d \bmod N$$

où la fonction de redondance $R(m)$ est donnée par:

$$R(m) = \omega \cdot m + a \quad \text{où} \quad \begin{cases} w \text{ est la redondance multiplicative} \\ a \text{ est la redondance additive} \end{cases} \tag{2}$$

La taille de la redondance est alors la taille de $R(m)$ moins la taille de m. Une redondance fixée à gauche $P\|m$ correspond donc à $\omega = 1$ et $a = P \cdot 2^{\ell}$, tandis qu'une redondance fixée à droite $m\|P$ s'obtient avec $\omega = 2^{\ell}$ et $a = P$.

L'attaque de De Jonge et Chaum [34], basée sur l'algorithme d'Euclide, permet de forger des signatures lorsque la taille de la redondance est inférieure au tiers de la taille de N, pour $\omega = 1$, ou lorsque cette taille est inférieure à la moitié de la taille de N si $a = 0$.

Cette attaque a été étendue par Girault et Misarsky [41], en utilisant un algorithme dû à Okamoto et Shiraishi [66], une extension de l'algorithme d'Euclide. L'attaque s'applique lorsque la taille de la redondance est inférieure à la moitié de la taille de N, pour toute valeur de ω et a. L'attaque s'applique aussi lorsqu'une redondance modulaire est utilisée (de la forme $m \bmod r$, pour un r fixé).

L'attaque de Misarsky [58] étend l'attaque précédente au cas où le message m et la redondance modulaire sont séparés en plusieurs parties, qui n'ont pas nécessairement la même taille:

$$R(m) = \sum_{i=1}^{k_1} m_i \cdot \omega_i + \sum_{j=1}^{k_2} (m \bmod r_j) \cdot \omega_{k_1+j} + a$$

L'attaque de Misarsky utilise l'algorithme de réduction de réseau LLL [52].

Finalement, l'attaque de Girault et Misarsky a été étendue par E. Brier, C. Clavier, J.S. Coron et D. Naccache [13] pour les signatures RSA à redondance affine $R(m) = \omega \cdot m + a$, avec une taille de redondance allant jusqu'aux deux-tiers de la taille du module; cela reste la meilleure borne connue à ce jour.

Autres attaques multiplicatives. L'attaque de Desmedt-Odlyzko [35] est une généralisation de l'attaque par masquage décrite précédemment. Il s'agit aussi d'une attaque à chiffré choisi sur le chiffrement RSA, ou de manière équivalente une attaque à message choisi sur la signature RSA. Dans une première phase, l'attaquant demande le déchiffrement d'un certain nombre de chiffrés appartenant à un certain ensemble S constitué de petits nombres premiers ainsi que d'autres entiers bien choisis. Ensuite, pour déchiffrer un chiffré c, l'attaquant essaye de factoriser dans la base S l'entier c ou bien $c \cdot r^e \bmod N$, pour un aléa r. Si cela est possible, alors l'attaquant peut retrouver le message clair correspondant à c grace aux propriétés multiplicatives de RSA.

L'attaque de Kaliski-Robshaw [50] est une variante de l'attaque précédente. Elle s'applique au cas où le message à déchiffrer est le produit de petits facteurs premiers pour le chiffrement RSA, et au cas où le message à signer est le produit de petits facteurs premiers pour la signature RSA.

1. Soit B un entier positif et $E = \{p : p \text{ premier tel que } p \leq B\}$.
2. Obtenir le déchiffrement (ou de manière équivalente la signature) de plusieurs messages dont tous les facteurs premiers sont inférieurs à B.
3. Par combinaison multiplicative, obtenir le déchiffrement (ou la signature) de tous les éléments de E.
4. L'attaquant peut ensuite déchiffrer (ou signer) tout message dont les facteurs premiers sont tous inférieurs à B.

L'attaque de Kaliski-Robshaw peut donc s'appliquer au cas des signatures RSA avec fonction de hachage lorsque la taille du haché est suffisamment petite. En effet, la probabilité qu'un entier se décompose en produit de petits facteur premier décroît de manière exponentielle avec sa taille. Il faut donc que la taille du haché soit suffisamment petite pour que cette probabilité ne soit pas trop faible.

Les attaques de Desmedt-Odlyzko et Kaliski-Robshaw ont été étendues par J.S. Coron, D. Naccache et J.P. Stern [30] au cassage du standard ISO 9796-2 [3] et d'une variante du standard ISO 9796-1 [2]. L'attaque a été étendue au standard ISO 9796-1 complet par D. Coppersmith, S. Halevi et C. Jutla [21] (voir aussi [26] et [19] pour une synthèse de ces attaques). A la suite de ces attaques, le standard ISO 9796-2 a été modifié, et le standard ISO 9796-1 a été abandonné.

Attaque contre RSA avec exposant privé petit. Pour réduire le temps nécessaire au déchiffrement d'un message ou à la génération d'une signature, on peut être tenté d'utiliser un exposant d petit. En effet, la durée d'une exponentiation modulaire étant proportionnelle à $\log_2 d$, si on prend un exposant d de 100 bits pour un module RSA de 1024 bits, on gagnera pratiquement un facteur 10 en temps de calcul. Malheureusement, on ne peut pas utiliser un exposant trop petit, comme le montre le théorème suivant dû à Wiener [75].

Théorème 1. *Soit $N = p \cdot q$ avec $q < p < 2 \cdot q$. Soit $d \leq \frac{1}{3} N^{1/4}$. Etant donné (N, e) avec $e \cdot d = 1 \bmod \phi(N)$, il existe un algorithme polynomial permettant de retrouver d.*

Nous verrons au chapitre 4 que l'attaque précédente a été amélioré en 1999 par Boneh et Durfee [10] pour un exposant $d < N^{0.292}$, en utilisant une variante du théorème de Coppersmith.

Attaques avec exposant public e petit. L'utilisation d'un exposant de chiffrement petit permet d'accélérer le chiffrement ou la vérification de signature. Il est possible de prendre $e = 3$, mais il est recommandé pour éviter certaines attaques de prendre $e = 2^{16} + 1 = 65537$. Dans ce cas, avec un module RSA de 1024 bits, on obtient un gain en temps proche d'un facteur 50 par rapport à un exposant e de pleine taille. Contrairement aux attaques contre les exposants de déchiffrement petits, les attaques avec e petit ne conduisent pas à un cassage total de RSA. Ces attaques, qui sont basées sur l'utilisation du théorème de Coppersmith [18], seront décrites en détail au chapitre 4.

L'attaque de Franklin-Reiter sur les messages liés. L'attaque de Franklin-Reiter sur les messages liés avec un e petit est la suivante [20]: supposons que deux messages m_1, m_2 vérifient une relation polynomiale connue p de la forme

$$m_2 = p(m_1), \qquad \deg(p) = \delta$$

et que les deux chiffrés correspondant $c_1 = m_1^e \bmod N$ et $c_2 = m_2^e \bmod N$ sont connus. On voit que dans ce cas $z = m_1$ est une racine commune des deux équations polynomiales:

$$z^e - c_1 = 0 \bmod N$$
$$p(z)^e - c_2 = 0 \bmod N$$

de sorte qu'avec forte probabilité on retrouve m_1 avec :

$$\text{PGCD}\,(z^e - c_1, p(z)^e - c_2) = z - m_1 \bmod N\ .$$

La complexité de l'attaque est quadratique en l'exposant e. Elle ne s'applique donc que lorsqu'un petit exposant e est utilisé.

L'attaque de Håstad. Supposons qu'un attaquant obtienne le chiffré d'un même message m pour trois clef publiques différentes, avec $e = 3$, soit :

$$c_1 = m^3 \bmod N_1 \quad c_2 = m^3 \bmod N_2 \quad c_3 = m^3 \bmod N_3$$

Alors on voit qu'en appliquant le théorème du reste Chinois à c_1, c_2, c_3, on obtient un entier $c' < N_1 \cdot N_2 \cdot N_3$ tel que :

$$c' = m^3 \bmod N_1 \cdot N_2 \cdot N_3$$

ce qui donne en fait $c' = m^3$ dans $\mathbb{Z}$ et permet donc de retrouver m. On voit que l'attaque se généralise à tout exposant de chiffrement e, à condition que le nombre de chiffrés obtenus soit supérieur ou égal à e. L'attaque n'est donc réalisable que pour e petit.

L'attaque de Håstad est une généralisation de l'attaque précédente au cas où les chiffrés sont de la forme :

$$c_i = f_i(m)^{e_i} \bmod N_i$$

où les f_i sont des polynômes publics. Le théorème suivant donne une version améliorée par D. Boneh [9] du théorème de Håstad et montre que l'on peut retrouver le message m sous certaines conditions.

Théorème 2. *Soient $N_1, \ldots, N_k$ des entiers premiers entre eux deux à deux. Soit $N_{min} = \min(N_i)$. Soient les k polynômes $g_i \in \mathbb{Z}_{N_i}[x]$ de degré maximum δ. On suppose qu'il existe un unique $m < N_{min}$ satisfaisant:*

$$g_i(m) = 0 \bmod N_i \quad \textit{pour tout } i = 1, \ldots, k$$

Alors si $k \geq \delta$, on peut retrouver efficacement m à partir des N_i et des polynômes g_i.

L'attaque de Bleichenbacher. L'attaque de Bleichenbacher est une attaque à chiffré choisi contre le standard de chiffrement PKCS#1 v1.5. Ce standard est défini de la façon suivante : à partir du message m et d'un entier aléatoire r, on forme:

$$\text{PKCS}(m, r) = 0002_{16} \| r \| 00_{16} \| m$$

et le chiffré c s'obtient en calculant:

$$c = \text{PKCS}(m, r)^e \bmod N$$

La notion d'attaque à chiffré choisi signifie que l'attaquant peut obtenir le déchiffrement de plusieurs chiffrés de son choix avant de devoir calculer le déchiffrement d'un c' donné. En fait, l'attaque de Bleichenbacher ne nécessite pas la connaissance complète du message

clair mais seulement la conformité ou non du chiffré avec le schéma d'encodage PKCS#1 v1.5. Autrement dit, l'attaquant a seulement accès à un oracle qui répond "oui" si au chiffré c correspond un clair m dont le format est conforme à PKCS#1 v1.5, et "non" dans le cas contraire. C'est une hypothèse réaliste dans la pratique: en effet, avant que cette attaque ne soit connue, de nombreux serveurs vérifiaient qu'un chiffré était PKCS#1 v1.5-conforme, et renvoyaient un message d'erreur dans le cas contraire. L'attaque de Bleichenbacher permettait alors de déchiffrer n'importe quel message à l'aide d'environ un million d'appel à cet oracle. En conséquence, le standard PKCS#1 v1.5 a été remplacé dans la version 2.0 [70] par le schéma OAEP, développé par Bellare et Rogaway [7], dont nous étudierons les propriétés de sécurité au chapitre 3.

3 Conclusion

Les attaques décrites dans ce chapitre ne mettent pas en cause l'algorithme RSA lui-même, mais plutôt son utilisation incorrecte. Nous verrons au chapitre 2 que de manière générale, l'approche correcte en cryptographie consiste à d'abord formaliser la notion de sécurité que l'on veut atteindre[1], pour ensuite construire un schéma qui satisfait cette notion de sécurité sous une hypothèse de complexité bien définie.

[1] Pour un schéma de signature, ce sera ce qu'on appelle la "résistance contre la forge existentielle lors d'une attaque à message choisie adaptative". Pour un schéma de chiffrement, ce sera la "sécurité sémantique sous une attaque à chiffré choisi adaptative".

Les notions de sécurité

1 Introduction

Le but de la cryptographie est de construire des schémas qui accomplissent un certain objectif malgré la présence d'un attaquant. Pour formaliser la sécurité d'un schéma, il faut donc spécifier ce que l'attaquant a le droit de faire, et sous quelles conditions son attaque réussit. Le schéma sera alors considéré comme "sûr" si on peut démontrer qu'une telle attaque est impossible (sauf éventuellement avec probabilité négligeable).

Par exemple, pour un schéma de chiffrement à clef publique, on pourrait considérer que l'attaquant doit retrouver la clef privée à partir de la clef publique. Mais il s'agit d'un objectif très ambitieux, qui correspond en fait à un cassage total du schéma. En réalité, l'attaquant pourrait poursuivre un objectif plus modeste, par exemple arriver à retrouver le message clair correspondant à un chiffré donné, ou même n'obtenir qu'un seul bit d'information sur ce clair.

Formaliser correctement la sécurité d'une fonctionnalité cryptographique n'est pas une tâche aisée. Pour les schémas de signature, la notion de sécurité satisfaisante n'a été obtenue qu'en 1988[1], et pour les schémas de chiffrement à clef publique, elle n'a été obtenue qu'en 1991[2], soit plus de treize ans après l'invention de RSA. Pourtant, la définition de la sécurité revêt une importance fondamentale : on ne peut pas espérer obtenir un schéma sûr si on ne sait pas ce que c'est que la sécurité ! Dans ce chapitre, nous rappelons comment se formalise la sécurité d'un schéma de signature et la sécurité d'un schéma de chiffrement à clef publique.

Une fois que la sécurité a été définie, l'étape suivante consiste à essayer de construire un schéma qui satisfait cette définition, de façon prouvée. La preuve prend généralement la forme d'une réduction : on prouve que si la notion de sécurité n'était pas satisfaite, alors il serait possible de résoudre efficacement un problème jugé difficile, comme par exemple la factorisation d'un module RSA. La sécurité d'un schéma n'est donc généralement pas absolue mais relative : on montre que le schéma est sûr sous une certaine hypothèse de complexité (par exemple, factoriser est difficile). Le domaine de la *sécurité prouvée* est la combinaison de ces trois étapes : définition, schéma, preuve. C'est cette approche qui est maintenant dominante dans la recherche en cryptographie.

Le *modèle de l'oracle aléatoire* a été introduit par Bellare et Rogaway en 1993 comme un paradigme permettant la construction de protocoles efficaces [6]. C'est un modèle d'exécution dans lequel les fonctions de hachage sont remplacées par un oracle aléatoire H auquel tout le monde a accès, y compris l'attaquant. Un oracle aléatoire est une fonction $H : \{0,1\}^* \to \{0,1\}^k$ qui a chaque entrée associe une sortie aléatoire dans $\{0,1\}^k$. Dans ce modèle, l'attaquant ne peut pas calculer le résultat de la fonction de hachage par lui-même, il doit faire appel l'oracle aléatoire. La sécurité du cryptosystème est alors prouvée dans ce modèle. Ensuite, pour utiliser concrètement le schéma, on instancie l'oracle aléatoire avec une certaine fonction de hachage cryptographique. C'est cette dernière étape qui est

[1] C'est la "résistance contre la forge existentielle lors d'une attaque à message choisie adaptative" [43].
[2] C'est la "sécurité sémantique sous une attaque à chiffré choisi adaptative" [68].

heuristique : en effet, une preuve dans le modèle de l'oracle aléatoire n'implique pas que le schéma reste sûr lorsqu'on utilise à la place une fonction de hachage bien définie.

Le modèle de l'oracle aléatoire s'est révélé très utile pour obtenir des cryptosystèmes plus simples et plus efficaces que ceux prouvés sûrs dans le modèle standard (par exemple, les schémas OAEP [7] pour le chiffrement et PSS [8] pour la signature). D'un point de vue théorique, il est clair qu'une preuve dans le modèle de l'oracle aléatoire ne fournit qu'une garantie heuristique lorsqu'on l'instancie avec une fonction de hachage particulière comme par exemple SHA-1 [64]. Le modèle de l'oracle aléatoire a ainsi été sujet à beaucoup de controverses, surtout depuis l'article de Canetti, Goldreich et Halevi [14] publié en 1998 exhibant un schéma sûr dans le modèle de l'oracle aléatoire, mais dont l'instanciation dans le modèle standard sera toujours vulnérable. De nombreux autres résultats de séparation ont suivis [61, 45, 54], mettant en évidence différents schémas cryptographiques surs dans le modèle de l'oracle aléatoire mais vulnérables pour toute instanciation concrète de l'oracle aléatoire (et cela même avec une *famille* de fonction de hachage). Cependant, ces résultats de séparation, bien qu'importants sur le plan théorique, ne semblent pas constituer une menace directe contre les schémas couramment utilisés basés sur l'oracle aléatoire (comme OAEP [7] et PSS [8], utilisés dans le standard PKCS #1 v2.1 [70]).

Nous reviendrons dans la suite du chapitre sur le niveau de sécurité donné par une preuve dans le modèle de l'oracle aléatoire. Nous verrons en particulier comment construire une fonction de hachage permettant d'émuler un oracle aléatoire, à partir d'une fonction de compression ou d'un block-cipher supposés idéaux. Ce résultat, obtenu conjointement avec Yevgeniy Dodis, Cécile Malinaud et Prashant Puniya, a été publié à la conférence Crypto 2005 [27]; l'article est reproduit en annexe.

2 Sécurité des schémas de chiffrement et des schémas de signature

2.1 La sécurité des schémas de signature

La signature d'un message est une chaîne de bits qui dépend du message et d'un secret que seul le signataire connaît. Une signature numérique doit être vérifiable: n'importe qui doit pouvoir vérifier la validité de la signature.

Définition 1 (schéma de signature). *Un schéma de signature est défini de la façon suivante:*

- L'algorithme de génération de clef `Gen` *est un algorithme probabiliste qui étant donné* 1^k*, renvoie une paire* (pk, sk) *de clef publique* pk *et privée* sk *correspondantes.*

- L'algorithme de signature `Sign` *prend en entrée un message* m *et la clef privée* sk *et retourne la signature* $x = \mathtt{Sign}_{sk}(m)$*. L'algorithme de signature peut être probabiliste.*

- L'algorithme de vérification `Verify` *prend en entrée un message* m*, la signature candidate* x' *et la clef publique* pk*. Il renvoie un bit* $\mathtt{Verify}_{pk}(M, x')$*, égal à* 1 *si la signature est acceptée, et* 0 *sinon. On requiert que si* $x \leftarrow \mathtt{Sign}_{sk}(M)$*, alors* $\mathtt{Verify}_{pk}(M, x) = 1$.

La notion de sécurité la plus forte pour les schémas de signature a été définie par Goldwasser, Micali et Rivest dans [43]: il doit être impossible pour un attaquant de produire une signature valide d'un message quelconque, même en ayant la possibilité d'obtenir la signature de messages de son choix. On peut définir cette notion de sécurité en considérant le scénario suivant entre un challengeur et un attaquant $\mathcal{A}$:

Initialisation : Le challengeur exécute l'algorithme `Gen` pour obtenir une clef publique pk et la privée sk correspondante. L'attaquant $\mathcal{A}$ reçoit pk.

Requêtes : L'attaquant $\mathcal{A}$ requiert les signatures d'au plus q_s messages de son choix $m_1, \ldots, m_{q_s} \in \{0,1\}^*$, sous la clef pk, de manière adaptative. Le challengeur répond pour chaque requête avec une signature $\sigma_i = \mathtt{Sign}_{sk}(m_i)$.

Sortie : A la fin, $\mathcal{A}$ renvoie une paire message/signature (m, σ) et réussit l'attaque si m est différent de chacun des m_i, et si $\mathtt{Verify}_{pk}(m, \sigma) = 1$.

On définit $\mathrm{AdvSig}_{\mathcal{A}}$ comme la probabilité que $\mathcal{A}$ réussisse l'attaque lors du scénario précédent, la probabilité étant prise sur les aléas du challengeur et de l'attaquant.

Définition 2. *On dit qu'un forgeur $\mathcal{A}$ (t, q_s, ε)-casse un schéma de signature si $\mathcal{A}$ s'exécute en un temps au plus t, s'il requiert au plus q_s signatures, et si $AdvSig_{\mathcal{A}}$ est au moins égal à ε. Un schéma de signature est dit (t, q_s, ε)-sûr s'il n'existe pas de forgeur qui (t, q_s, ε)-casse ce schéma de signature.*

Lorsque la sécurité du schéma est prouvée dans le modèle de l'oracle aléatoire, on ajoute un quatrième paramètre q_h qui donne le nombre maximal de requêtes que l'attaquant peut faire à l'oracle aléatoire. Le schéma PSS [8] est un exemple de schéma sûr dans le modèle de l'oracle aléatoire; nous rappellerons la définition de ce schéma au chapitre 3.

2.2 La sécurité du chiffrement à clef publique

Le chiffrement à clef publique permet à quiconque connaissant la clef publique d'Alice de lui envoyer un message chiffré qu'elle seule sera en mesure de déchiffrer, à l'aide de sa clef privée.

Définition 3 (Chiffrement à clef publique). *Un schéma de chiffrement à clef publique est défini de la façon suivante:*

- L'algorithme de génération de clef `Gen` *est un algorithme probabiliste qui étant donné 1^k, renvoie une paire (pk, sk) de clef publique pk et privée sk correspondantes.*

- L'algorithme de chiffrement $\mathcal{E}$ prend en entrée un message m et la clef publique sk et renvoie un message chiffré $c = \mathcal{E}_{pk}(m)$. Cet algorithme de chiffrement peut être probabiliste.

- L'algorithme de déchiffrement $\mathcal{D}$ prend en entrée un chiffré c et la clef privée sk, et renvoie $m = \mathcal{D}_{sk}(c)$ ou bien $\perp$ dans le cas d'un chiffré non valide. Cet algorithme est déterministe. On requiert que $\mathcal{D}_{sk}(\mathcal{E}_{pk}(m)) = m$ avec probabilité 1.

La *sécurité sémantique*, définie par Goldwasser et Micali [42], formalise l'intuition selon laquelle un attaquant ne devrait pas pouvoir obtenir une quelconque information sur un message étant donné son chiffré. Cependant, cette garantie n'est valable que lorsque l'attaquant est complètement passif, c'est à dire lorsqu'il se contente de lire les messages transmis sans les modifier. La notion de sécurité sémantique à elle seule ne donne aucune garantie de secret lorsque l'attaquant peut mener une attaque active, c'est à dire lorsqu'il peut modifier ou injecter des messages dans un réseau.

Pour prendre en compte les attaques actives, Rackoff et Simon [68] ont introduit la notion de *sécurité contre les attaques à chiffré choisi adaptative*. En menant un attaque

active, un attaquant peut par exemple injecter des messages chiffrés dans un réseau, et obtenir ainsi une information partielle sur les clairs correspondants. Rackoff et Simon modélisent ce type d'attaque en permettant à l'attaquant d'obtenir le déchiffrement de chiffrés de son choix; autrement dit, l'attaquant a accès à un oracle de déchiffrement. Ensuite, étant donné un chiffré c, nous voulons avoir la garantie que l'attaquant ne peut obtenir aucune information partielle sur le clair correspondant. On voit qu'il faut bien restreindre le comportement de l'attaquant, car sinon il pourrait tout simplement soumettre le chiffré c lui-même à l'oracle de déchiffrement. La restriction proposée par Rackoff et Simon est la plus faible qui soit : l'attaquant n'a pas le droit de soumettre le chiffré c à l'oracle de déchiffrement; en revanche, il a le droit de soumettre tout autre chiffré de son choix, même lié au chiffré c. Cette notion de sécurité est appelée IND-CCA2 dans [5]; elle est équivalente à la notion de *non-malléabilité* introduite par Dolev, Dwork et Naor [37, 5] (appelée NM-CCA2 dans [5]).

La notion de sécurité contre le attaques à chiffré choisi adaptative peut se définir en considérant le scénario suivant entre un challengeur et un attaquant $\mathcal{A}$:

Initialisation : Le challengeur exécute l'algorithme `Gen` pour obtenir une clef publique pk et la privée sk correspondante. L'attaquant $\mathcal{A}$ reçoit pk.

Requêtes avant challenge : L'attaquant requiert le déchiffrement d'une série de chiffrés, de manière adaptative. Le challengeur lui renvoie le clair correspondant en utilisant la clé privée sk.

Challenge : L'attaquant prépare deux messages m_0 et m_1 de son choix, et les envoie au challengeur. Le challengeur génère un bit b aléatoire, et renvoie $y = \mathcal{E}_{pk}(m_b)$ à l'attaquant.

Requêtes après challenge : L'attaquant continue de requérir le déchiffrement d'une série de chiffrés, de manière adaptative. La seule restriction est que ces chiffrés doivent être différents de y.

Sortie : L'attaquant renvoie un bit b', représentant son "estimation" de b.

Ceci termine la description du scénario de l'attaque. On définit l'avantage de l'attaquant $\mathcal{A}$ comme :

$$\mathrm{Adv}(\mathcal{A}) = |\Pr[b' = b] - 1/2|$$

où la probabilité est prise sur les aléas du challengeur et de l'attaquant.

Définition 4. *On dit qu'un schéma de chiffrement à clef publique est (t, q_d, ε)-sûr contre les attaques à chiffré choisi adaptatives si pour tout attaquant $\mathcal{A}$ s'exécutant en un temps au plus t et requérant au plus q_d déchiffrements, $Adv(\mathcal{A})$ est au plus égal à ε.*

Lorsque la sécurité du schéma est prouvée dans le modèle de l'oracle aléatoire, on ajoute un quatrième paramètre q_h qui donne le nombre maximal de requêtes que l'attaquant peut faire à l'oracle aléatoire. Il existe aussi des notions de sécurité intermédiaires entre la sécurité sémantique et la sécurité contre les attaques à chiffré choisi. Naor et Yung ont proposé un modèle d'attaque [59] dans lequel l'attaquant a seulement accès à l'oracle de déchiffrement *avant* d'avoir obtenu le challenge y. Cette notion de sécurité est appelée IND-CCA1 dans [5].

Nous verrons au chapitre 3 des exemples de schémas sûrs contre les attaques à chiffré choisi adaptatives. Un exemple de schéma prouvé sûr dans le modèle de l'oracle aléatoire

est le schéma OAEP [7], sous l'hypothèse de la *one-wayness* de la permutation sous-jacente, même sur un domaine partiel [38] (cette hypothèse de complexité sera définie au chapitre 3). Le premier schéma de chiffrement à clé publique à la fois utilisable dans la pratique et sûr dans le modèle standard a été proposé par Cramer et Shoup en 1998 [32].

3 Nouvelle notion de sécurité pour les fonctions de hachage

Comme nous l'avons rappelé en introduction, le modèle de l'oracle aléatoire est un modèle d'exécution dans lequel la fonction de hachage H est vue comme un oracle aléatoire. On peut en se demander s'il est raisonnable de modéliser ainsi une fonction de hachage. A première vue, il semble qu'on ne puisse rien dire de significatif sur cette question : on sait tout simplement qu'une fonction de hachage concrètement définie H n'est pas un oracle aléatoire. Cependant, si dans le modèle de l'oracle aléatoire on modélise H comme une fonction de $\{0,1\}^*$ vers $\{0,1\}^n$ (où n est proportionnel au paramètre de sécurité), dans la pratique, pour construire une telle fonction prenant en entrée une chaîne de taille arbitraire, on commence d'abord par construire une primitive prenant une entrée de taille fixe (une fonction de compression ou un block-cipher), que l'on itère ensuite pour étendre le domaine de l'entrée sur une taille arbitraire. Ainsi, la plupart des fonctions de hachage (comme par exemple SHA-1 et MD5) sont construites en utilisant la construction dite de Merkle-Damgård (voir la figure 1):

Fonction $H(m_1, \ldots, m_\ell)$:
soit $y_0 = 0^n$ (plus généralement, on peut utiliser n'importe quelle constante IV)
pour $i = 1$ à ℓ faire $y_i \leftarrow f(y_{i-1}, m_i)$
retourner y_ℓ

où f est une fonction de compression $f : \{0,1\}^{n+\kappa} \to \{0,1\}^n$. Si le nombre de blocks ℓ n'est pas fixé, on ajoute un block supplémentaire $m_{\ell+1}$ contenant la représentation binaire $\langle |m| \rangle$ de la longueur du message. La fonction f peut être construite à partir de rien ou bien réalisée à l'aide d'un block-cipher E avec la construction de Davies-Meyer $f(x,y) = E_y(x) \oplus x$. Par exemple, la fonction de compression de SHA-1 a été construite spécifiquement pour le hachage, mais on peut néanmoins en dériver un block-cipher (voir [46]).

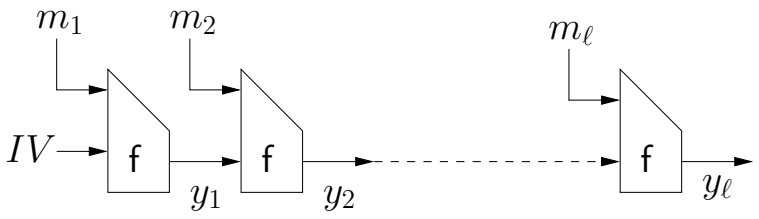

Fig. 1. La construction de Merkle-Damgård classique.

On voit donc que les fonctions de hachage utilisées dans la pratique ont une structure itérative bien particulière, très différente du caractère "monolithique" de l'oracle aléatoire.

Autrement dit, même si on suppose que la primitive f est idéale (nous verrons comment cela peut se formaliser), une preuve dans le modèle de l'oracle aléatoire ne garantit pas la sécurité du schéma lorsqu'une telle fonction itérative H est utilisée. On peut illustrer ce point avec un exemple bien connu. On pourrait imaginer de construire un algorithme de MAC en incluant simplement la clef secrète k dans l'entrée de la fonction de hachage, en prenant pas exemple $\mathrm{MAC}(k, m) = H(k\|m)$. Il est facile de voir que cette construction est sure lorsqu'on modélise H comme un oracle aléatoire, car aucun attaquant ne peut produire une forge sauf avec probabilité négligeable. Cependant, il est facile de voir que ce schéma est complètement vulnérable lorsqu'on utilise une construction de type Merkle-Damgård, quelle que soit la fonction de compression f utilisée. En effet, étant donné $\mathrm{MAC}(k, m) = H(k\|m)$, on peut étendre le message m avec un block arbitraire y et en déduire $\mathrm{MAC}(k, m\|y) = H(k\|m\|y)$ sans connaître la clé k. On voit donc que la construction d'un MAC à partir d'une fonction de hachage itérative requiert une analyse spécifique, et ne peut pas se déduire de la sécurité du même MAC avec une fonction de hachage H monolithique.

Pour réduire cet écart entre d'une part le modèle de l'oracle aléatoire et d'autre part son instanciation par une fonction de hachage itérative, nous proposons plusieurs constructions de fonctions de hachage itératives qui "émulent" un oracle aléatoire à partir d'une brique de base (fonction de compression f ou block-cipher E) supposée idéale. Nous donnons une définition formelle ce de que signifie "émuler" dans ce contexte. Nous montrons que lorsqu'une construction particulière de H à partir de f (ou de E) satisfait cette définition, alors n'importe quel schéma prouvé sûr en modélisant H comme un oracle aléatoire restera sûr si l'on utilise cette construction particulière à la place (en modélisant f ou E comme une primitive idéale). Nous avons déjà vu que la construction de Merkle-Damgård (sur laquelle est basée SHA-1) ne satisfait pas cette définition. Nous proposons donc plusieurs constructions qui permettent de satisfaire cette définition; ces constructions n'introduisent que très peu de changement par rapport à la construction de Merkle-Damgård classique, et sont donc facilement implémentables dans la pratique. Ces résultats, obtenus conjointement avec Yevgeniy Dodis, Cécile Malinaud et Prashant Puniya, ont été publiés à la conférence Crypto 2005 [27]; l'article est reproduit en annexe.

Schémas de chiffrement et de signature à clef publique

1 Introduction

Nous avons rappelé au chapitre 2 les notions de sécurité pour le chiffrement à clef publique et pour la signature. Nous avons vu que les preuves de sécurité sont le plus souvent relatives : on montre que si un schéma ne vérifie pas une certaine notion de sécurité, alors il est possible de résoudre un problème mathématique supposé difficile, comme par exemple la factorisation d'un module RSA. La sécurité des schémas cryptographiques est donc le plus souvent basée sur une certaine hypothèse de complexité. Nous commençons donc par rappeler les principales hypothèses de complexité utilisées en cryptographie. Nous rappelons ensuite les principales constructions connues pour le chiffrement et pour la signature.

2 Les hypothèses de complexité

2.1 Le problème RSA

Soit un entier N, produit de deux nombres premiers aléatoires p et q de taille k bits, et soit e un entier aléatoire tel que $1 < e < \phi(N)$ et $\gcd(e, \phi(N)) = 1$. Nous avons vu au chapitre 1 que la fonction de chiffrement RSA était définie de la façon suivante :

$$f : \mathbb{Z}_N^* \to \mathbb{Z}_N^*,\ x \to x^e \bmod N$$

Le problème RSA peut alors se formuler de la façon suivante :

Définition 1 (Problème RSA). *Soit (N, e) générés tels que précédemment, pour un paramètre entier k. Soit y un entier aléatoire dans $\mathbb{Z}_N$. Étant donné (N, e, y), calculer x tel que $x^e = y \bmod N$*

L'*hypothèse RSA* suppose donc que le problème RSA est difficile, c'est à dire qu'étant donné (N, e, y) générés comme précédemment, il n'existe pas d'algorithme de complexité polynomiale en k qui renvoie x avec probabilité non négligeable en fonction de k [1]. On peut aussi considérer le problème RSA avec petit exposant e en fixant un e petit au lieu de le générer aléatoirement (par exemple, en prenant e polynomial en k).

On ne sait pas si le problème RSA est équivalent au problème de la factorisation. Il existe d'ailleurs des arguments qui indiquent le contraire [12], pour un exposant e petit. Cependant, le problème RSA peut fort bien être un problème difficile sans pour autant être équivalent à la factorisation.

Le problème RSA *sur un domaine partiel* a été introduit Fujisaki, Okamoto, Pointcheval et Stern dans [38] pour prouver la sécurité du schéma de chiffrement OAEP [7]. Dans cette variante, on ne demande pas de renvoyer la pré-image complète de y mais seulement une partie de cette pré-image.

[1] On dit qu'une fonction $g : \mathbb{N} \to \mathbb{R}$ est *négligeable* si pour tout $c \in \mathbb{N}$, il existe $k_0 \in \mathbb{N}$ tel que pour tout $k > k_0$, $g(k) < k^{-c}$.

Définition 2 (Problème RSA sur un domaine partiel). *Soit (N, e) générés tels que précédemment, pour un paramètre entier k. Soit k_1 un paramètre fonction de k. Soit y un entier aléatoire dans $\mathbb{Z}_N$ et soit x tel que $x^e = y \bmod N$. On écrit $x = \omega \| s$, avec $\omega \in \{0,1\}^{k_1}$. Étant donné (N, e, y), renvoyer ω.*

On peut en fait généraliser cette variante pour toute fonction à trappe f. Le problème de l'inversion de f sur un domaine partiel est toujours plus facile que le problème de l'inversion complète de f. Cependant, pour la fonction RSA, on peut montrer que les deux problèmes sont équivalents [38, 28], en utilisant l'algorithme de réduction de réseaux LLL [52].

Une autre variante du problème RSA est le problème RSA fort, introduit dans [4], pour lequel on n'impose pas le choix de e :

Définition 3 (Problème RSA fort). *Soit N généré comme précédemment, pour un paramètre entier k. Soit y un entier aléatoire dans $\mathbb{Z}_N$. Étant donné (N, y), renvoyer $e \geq 2$ et x tel que $x^e = y \bmod N$*

2.2 Le problème du logarithme discret

De nombreux schémas cryptographiques sont basés sur le problème du logarithme discret dans un groupe fini. Les groupes les plus souvent utilisés sont les sous-groupes multiplicatifs de $\mathbb{Z}_p^*$ ainsi que le groupe des points de certaines courbe elliptiques. On pense que le problème du logarithme discret est difficile sur ces groupes. Soit G un groupe cyclique d'ordre premier q, et soit g un générateur de G. On peut définir les problèmes suivants, par ordre décroissant de difficulté :

Définition 4 (Problème du Logarithme Discret). *Soit x un entier aléatoire dans $\mathbb{Z}_q$. Étant donné (g, g^x), calculer x.*

Définition 5 (Problème Diffie-Helmann). *Soient x, y deux entiers aléatoires dans $\mathbb{Z}_q$. Étant donné (g, g^x, g^y), calculer g^{xi}.*

Définition 6 (Problème Diffie-Helmann décisionnel). *Soient x, y, z trois entiers aléatoires dans $\mathbb{Z}_q$, et soit b un bit aléatoire. On définit $u = g^{xy}$ si $b = 0$, et $u = g^z$ sinon. Étant donné (g, g^x, g^y, u), renvoyer b.*

Pour le problème décisionnel, on définit l'*avantage* d'un attaquant $\mathcal{A}$ renvoyant un bit b' par :

$$\mathrm{Av}(\mathcal{A}) = |\Pr[b' = b] - 1/2|$$

L'hypothèse de la difficulté du logarithme discret consiste donc à supposer qu'il n'existe pas d'attaquant $\mathcal{A}$ de complexité polynomiale résolvant le problème du logarithme discret avec probabilité non-négligeable. L'hypothèse Diffie-Helmann est définie de façon analogue. Pour l'hypothèse Diffie-Helmann décisionnel, on doit supposer que $\mathrm{Adv}(\mathcal{A})$ est négligeable pour tout attaquant $\mathcal{A}$ de complexité polynomiale.

La difficulté du problème du logarithme discret et de ses variantes dépend évidemment du groupe sous-jacent. Le meilleure algorithme générique (c'est à dire n'utilisant pas la structure du groupe sous-jacent) a une complexité en $\sqrt{q}$. Le problème du logarithme

discret sur les sous-groupes multiplicatifs de $\mathbb{Z}_p^*$ peut se résoudre en temps sub-exponentiel à l'aide d'une variante de la méthode du crible algébrique utilisé pour la factorisation [51]. On doit donc utiliser un premier p de taille comparable à celle d'un module RSA, c'est à dire au moins 2048 bits pour des applications devant garder un niveau de sécurité élevé pendant plusieurs années.

3 Le chiffrement à clef publique

Le premier schéma de chiffrement sémantiquement sûr a été proposé en 1984 par Goldwasser et Micali [42]. La sécurité sémantique permet de garantir que l'attaquant ne peut pas obtenir d'information sur le clair à partir du chiffré. Cependant, nous avons vu au chapitre 2 que la notion de sécurité sémantique à elle seule ne donne aucune garantie de secret lorsque l'attaquant peut mener une attaque active, c'est à dire lorsqu'il peut obtenir le déchiffrement de messages de son choix.

Naor et Yung [59] ont présenté le premier schéma de chiffrement à clef publique prouvé sûr contre les attaques à chiffré choisis non-adaptatives (IND-CCA1). Un schéma de chiffrement à clef publique prouvé sûr contre les attaques à chiffré choisi adaptatives (IND-CCA2) a ensuite été proposé par Dolev, Dwork et Naor dans [37]. Rackoff et Simon ont aussi proposé un schéma de chiffrement sûr contre les attaques à chiffré choisi adaptatives [68], mais ce schéma requiert la présence d'un centre de confiance. L'inconvénient des schémas précédents est qu'ils sont difficiles à mettre en oeuvre (bien que de complexité polynomiale); en effet, ils sont basés sur des constructions générales de preuves non-interactives à divulgation nulle de connaissance (non-interactive zero-knowledge proofs) qui sont inefficaces dans la pratique.

Le premier schéma de chiffrement à clef publique qui soit à la fois efficace et prouvé sûr contre les attaques à chiffré choisi adaptatives dans le modèle standard (*i.e.*, sans oracle aléatoire) a été proposé par Cramer et Shoup [31] en 1998. La sécurité du schéma repose sur la difficulté du problème Diffie-Helmann décisionnel. La difficulté du problème Diffie-Helmann décisionnel est équivalente à la sécurité sémantique du schéma de chiffrement El-Gamal de base [39]. Le schéma de Cramer-Shoup permet donc d'atteindre le niveau de sécurité contre les attaques à chiffré choisi adaptatives, au prix de seulement quelques calculs supplémentaires.

Avec une sécurité dans le modèle de l'oracle aléatoire, le schéma OAEP (Optimal Asymmetric Encryption Padding), proposé par Bellare et Rogaway [7] en 1994, fut le premier schéma de chiffrement à clef publique efficace et sûr contre les attaques à chiffré choisi adaptatives. Le schéma OAEP est défini à partir d'une permutation à sens unique à trappe f (par exemple, la fonction RSA). Pour chiffrer un message m, on commence par générer un aléa r et on applique la transformation décrite à la figure 2, pour obtenir $s\|t$. Ensuite on applique la fonction à trappe f pour obtenir le chiffré $c = f(s\|t)$. Pour retrouver m à partir de c, on calcule $s\|t = f^{-1}(c)$ à l'aide de la clé privée. Cela permet de retrouver le message m, qui est retourné si la redondance 0^{k_1} est vérifiée. Le schéma OAEP fait partie du standard de chiffrement PKCS #1 v2.1 [70].

En fait, contrairement à ce que l'on pensait, le schéma OAEP n'atteint pas le niveau de sécurité IND-CCA2 sous la seule hypothèse de la one-wayness de la permutation f, comme cela a été montré par Shoup en 2001 [73]. Pour atteindre le niveau de sécurité IND-CCA2, il faut une hypothèse plus forte: l'hypothèse de la one-wayness de la permutation f, même

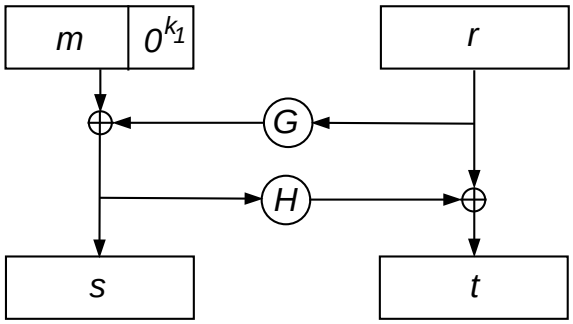

Fig. 2. Le schéma d'encodage OAEP

sur un domaine partiel [38]. Nous avons vu que pour la fonction RSA, le problème de la one-wayness est équivalent au problème de la one-wayness sur un domaine partiel; par conséquent, le schéma RSA-OAEP est sûr dans le modèle de l'oracle aléatoire, en supposant que le problème RSA est difficile.

Depuis la publication du schéma OAEP, de nombreuses constructions ont été proposées, qui permettent de convertir une fonction à sens unique à trappe en un schéma de chiffrement sûr contre les attaques à chiffré choisi adaptatives. Par exemple, le schéma REACT (Rapid Enhanced-security Asymmetric Cryptosystem Transform) [65] proposé par Okamoto et Pointcheval en 2001, permet d'utiliser une fonction à sens unique à trappe f qui n'est pas nécessairement une permutation.

Finalement, dans un article publié à la conférence Crypto 2002 [28], nous montrons que l'on peut utiliser le schéma d'encodage de PSS pour le chiffrement; l'article est joint en annexe. Le schéma d'encodage de PSS [8], proposé par Bellare et Rogaway en 1996, était originellement destiné à obtenir un schéma de signature basé sur RSA avec une preuve de sécurité dans le modèle de l'oracle aléatoire (voir Figure 3). Nous montrons dans cet article que l'encodage PSS peut aussi s'utiliser en chiffrement (à partir d'une permutation à sens unique à trappe comme RSA), et que l'on obtient ainsi un schéma de chiffrement à clef publique au niveau de sécurité IND-CCA2 dans le modèle de l'oracle aléatoire. On peut donc utiliser l'encodage PSS indifféremment pour le chiffrement ou la signature. On montre aussi que l'on peut utiliser dans ce cas la même clef publique pour le chiffrement et la signature (ce qui est d'habitude déconseillé). Cela permet de simplifier la gestion des clefs et l'implémentation des schémas à clef publique.

4 Schémas de signature

Le premier schéma de signature prouvé sûr contre la forger existentielle sous une attaque a message choisi a été proposé par Goldwasser, Micali et Rivest en 1988 dans [43]. Cependant, l'inconvénient de ce schéma est qu'il nécessite le maintient d'une variable d'état par le signeur, ce qui peut être un problème dans la pratique. Cramer et Shoup ont proposé en 1999 dans [32] un schéma de signature prouvé sûr contre les attaques à clair choisi adaptatives, qui est à la fois sans variable d'état et plus efficace. La sécurité du schéma est basée sur le problème RSA fort. Gennaro, Halevi et Rabin ont proposé dans [40] un

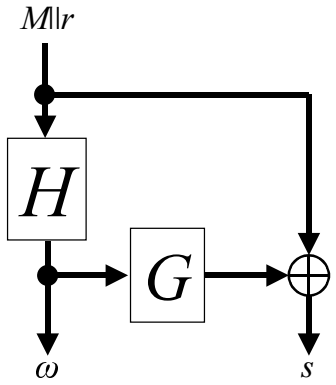

Fig. 3. Le schéma d'encodage PSS-R

nouveau schéma de signature de type hache-et-signe, prouvé sûr contre les attaques à clair choisi adaptatives, également sans variable d'état, efficace, et basé sur le problème RSA fort. On dit qu'un schéma de signature est de type hache-et-signe si le message est d'abord haché en utilisant une fonction de hachage, dont le résultat est ensuite signé en utilisant un schéma de signature comme RSA.

Dans le modèle de l'oracle aléatoire, Bellare et Rogaway ont montré dans [6] comment construire un schéma de signature prouvé sûr, à partir de n'importe quelle permutation à sens unique à trappe. Ils ont défini dans [8] le schéma de signature *Full Domain Hash* (FDH), qui est prouvé sûr dans le modèle de l'oracle aléatoire en supposant qu'inverser RSA est difficile, et le schéma *Probabilistic Signature Scheme* (PSS), qui a une meilleure preuve de sécurité que le schéma FDH. David Pointcheval et Jacques Stern ont prouvé dans [67] dans le modèle de l'oracle aléatoire la sécurité de certains schémas de signature basés sur le problème du logarithme discret.

Dans un article publié à la conférence Eurocrypt 2002 [24], nous décrivons une nouvelle preuve de sécurité pour le schéma de signature PSS, qui permet de réduire la taille de l'aléa utilisé lors de la génération de la signature et ainsi d'augmenter la bande passante au niveau du message transmis [24]; l'article est joint en annexe. On introduit aussi une technique permettant de montrer qu'une preuve de sécurité pour schéma de signature est optimale, et nous appliquons cette technique aux schémas FDH et PSS.

Dans un article publié à la conférence Crypto 2002 [25], nous étudions la sécurité des schémas de signature dans lesquels la taille de la fonction de hachage utilisée lors de l'encodage n'est qu'une fraction de celle du module; l'article est joint en annexe. Nous montrons que pour $e = 2$ (signatures Rabin), le schéma de signature est sûr dans le modèle de l'oracle aléatoire lorsque la taille du haché est au moins égale aux 2/3 de celle du module. Cela permet de prouver la sécurité des standards de signature ISO 9796-2 [3] et PKCS#1 v1.5 [72], à condition que la taille du haché soit suffisamment grande.

La réduction de réseaux et les techniques de Coppersmith

1 Introduction

Soient $u_1, \ldots, u_\omega \in \mathbb{Z}^n$ des vecteurs linéairement indépendants. On définit le *réseau* engendré par ces vecteurs comme l'ensemble des combinaisons linéaires à coefficients entiers de ces vecteurs, c'est à dire :

$$L = \{ \sum_{i=1}^{\omega} n_i \cdot u_i | \ n_i \in \mathbb{Z} \}$$

Un tel ensemble de vecteurs u_i forme une *base* du réseau. Toutes les bases d'un réseau ont le même nombre d'éléments ω, appelé la *dimension* ou le *rang* du réseau. On dit qu'un réseau est de rang plein si $\omega = n$.

L'algorithme LLL [52], publié par Lenstra, Lenstra et Lovász en 1982, permet d'obtenir une base d'un réseau avec des vecteurs relativement courts et orthogonaux. Cet algorithme a connu de nombreuses applications en cryptanalyse: cryptanalyse des schémas basés sur le problème du sac-à-dos [62], recherche de petites solutions de systèmes linéaires, attaque du schéma NTRU [48, 22] (voir [60] pour un synthèse très complète des applications de la réduction de réseau en cryptographie).

La recherche de petites racines de polynômes à une variable modulo un entier N de factorisation inconnue semble être un problème difficile dans le cas général. En effet, pour certains polynômes ce problème est équivalent à celui de la factorisation de N (par exemple, pour $p(x) = x^2 - a$). De plus, le problème de l'inversion de la fonction RSA est aussi un cas particulier de ce problème (avec $p(x) = x^e - a$). Cependant, à la conférence Eurocrypt 1996 [17, 18], Coppersmith a montré que le problème consistant à trouver les *petites* racines d'un polynôme pouvait se résoudre en temps polynomial, à l'aide de l'algorithme LLL :

Théorème 1 (Coppersmith). *Soit un polynôme $P(x)$ de degré δ, modulo un entier N de factorisation inconnue. Il existe un algorithme permettant de trouver tous les entiers x_0 tels que $P(x_0) = 0 \bmod N$ et $|x_0| \leq N^{1/\delta}$, en temps polynomial en $(\log N, \delta)$.*

L'algorithme précédent peut être étendu au cas des polynômes à plusieurs variables modulo un entier donné, mais cette extension est uniquement heuristique. L'algorithme de Coppersmith a trouvé de nombreuses applications en cryptographie, que nous rappelons dans la suite de ce chapitre: cryptanalyse de RSA avec petit exposant e lorsqu'une partie du message est connue [17], cryptanalyse de RSA lorsque l'exposant privé d est inférieur à $N^{0.29}$ [10], factorisation en temps polynomial de $N = p^r q$ pour r grand [11]. L'algorithme de Coppersmith permet aussi d'obtenir une preuve de sécurité améliorée pour le schéma de chiffrement RSA-OAEP avec petit exposant e [73].

Dans ce chapitre, nous présentons une nouvelle application de l'algorithme de Coppersmith pour montrer l'équivalence *déterministe* entre la connaissance de la clé privée de RSA et la factorisation du module. L'équivalence *probabiliste* est bien connue: il existe en effet un algorithme probabiliste dû à Miller [57] qui étant donné (N, e, d) renvoie la factorisation de N en temps polynomial. L'équivalence déterministe a été montrée par A.

May à la conférence Crypto 2004 [55]. L'algorithme a ensuite été simplifié et rendu plus efficace par J.S. Coron et A. May dans [29], dont l'article est reproduit en annexe.

Après sa publication, l'algorithme de Coppersmith a été simplifié par Howgrave-Graham [49]. Cette simplification s'étend elle aussi (heuristiquement) au cas des polynômes modulaires à plusieurs variables, et la sélection des paramètres permettant d'obtenir les meilleures bornes sur la taille des racines s'en trouve grandement simplifiée. C'est en fait cette approche qui est utilisée dans la plupart des attaques basées sur l'algorithme de Coppersmith (par exemple, [10, 11]).

Il existe un autre algorithme de Coppersmith, publié dans [16, 18], permettant de trouver les *petites* racines entières d'un polynôme à deux variables sur les entiers :

$$p(x,y) = \sum_{i,j} p_{i,j} \cdot x^i y^j$$

Dans le cas général, si on ne suppose pas que les racines sont petites, ce problème semble difficile car on voit que la factorisation est un cas particulier (en prenant $p(x,y) = x \cdot y - N$). Coppersmith a montré dans [16, 18] que l'on peut trouver efficacement les *petites* racines d'un polynôme à deux variables sur les entiers, toujours à l'aide de LLL :

Théorème 2 (Coppersmith). *Soit $p(x,y)$ un polynôme irréductible à deux variables à coefficients entiers, de degré maximum δ séparément en chaque variable. Soient X et Y deux bornes supérieures sur les solutions (x_0, y_0), et soit $W = \max_{i,j} |p_{ij}| X^i Y^j$. Si $XY < W^{2/(3\delta)}$, alors en temps polynomial en $(\log W, 2^\delta)$, on peut trouver toutes les paires d'entiers (x_0, y_0) telles que $p(x_0, y_0) = 0$, $|x_0| \leq X$ et $|y_0| \leq Y$.*

La borne sur la taille maximale de x_0 et y_0 dépend en fait de la forme du polynôme $p(x,y)$. Par exemple, pour un polynôme $p(x,y)$ de degré *total* δ en x et y, la borne devient $XY < W^{1/\delta}$. La technique peut aussi s'étendre de façon heuristique aux polynômes à plus de deux variables. Une application de ce deuxième algorithme de Coppersmith est la factorisation en temps polynomial d'un module RSA $N = pq$ lorsque la moitié des bits les plus significatifs ou les moins significatifs de p est connue [18].

Cependant, l'approche précédente pour le cas des polynômes à deux variables sur les entiers est assez difficile à comprendre et à implémenter [15]. En particulier, il est difficile de dériver les bornes sur la taille des racines obtenues en fonction de la forme du polynôme que l'on cherche à résoudre. L'analyse est plus complexe que pour les polynômes à une variable car les réseaux obtenus ne sont pas de rang plein, ce qui rend le calcul du déterminant plus difficile. Dans ce chapitre, nous présentons une variante plus simple de l'algorithme de Coppersmith pour trouver les petites racines d'un polynôme à deux variables sur les entiers. Cette variante a été publiée à la conférence Eurocrypt 2004 [23], dont l'article est reproduit en annexe. La simplification obtenue est analogue à celle développée par Howgrave-Graham pour le cas des polynômes modulaires à une variable. L'avantage de cette nouvelle approche est que le réseau obtenu est maintenant de rang plein, avec une base triangulaire naturelle. On peut alors obtenir immédiatement le déterminant du réseau et ainsi les bornes sur les racines en fonction de la forme du polynôme. L'inconvénient de ce nouvel algorithme est qu'il est moins efficace que l'algorithme de Coppersmith: en effet, notre algorithme a une complexité polynomiale si $XY < W^{2/3\delta - \varepsilon}$ pour tout $\varepsilon > 0$ fixé, alors que l'algorithme de Coppersmith requiert seulement $XY < W^{2/3\delta}$.

Dans ce chapitre, après avoir rappelé le fonctionnement général de l'algorithme LLL, nous décrirons en détail l'algorithme de Coppersmith pour trouver les petites racines de polynômes modulaires à une variable. Nous verrons ensuite les principales applications de l'algorithme de Coppersmith en cryptographie. Finalement, nous décrirons notre simplification du second algorithme de Coppersmith pour trouver les petites racines de polynômes à deux variables sur les entiers.

2 L'algorithme LLL

Soit $B = (u_1, \ldots, u_\omega) \in \mathbb{Z}^n$ une liste de ω vecteurs linéairement indépendants. Le réseau $L(B)$ engendré par B est l'ensemble des combinaisons linéaires à coefficients entiers de ces vecteurs :

$$L(B) = \left\{ \sum_{i=1}^{\omega} n_i \cdot u_i | \ n_i \in \mathbb{Z} \right\}$$

Une telle suite B est appelée base du réseau. Toutes les bases d'un réseau ont le même nombre d'éléments ω, appelé la *dimension* ou le *rang* du réseau. On dit qu'un réseau est de rang plein si $\omega = n$.

Un réseau de dimension ≥ 2 possède une infinité de bases, liées entre elles par une matrice unimodulaire (matrice à coefficient entier de déterminant ± 1). Toutes les bases ont donc le même déterminant de Gram $\det < u_i, u_j >_{i,j}$. On définit le *déterminant* (ou le *volume*) d'un réseau comme la racine carrée du déterminant de Gram. Dans le cas particulier d'un réseau de rang plein ($\omega = n$), ce déterminant est égal à la valeur absolue du déterminant de la matrice carrée donnée par les vecteurs u_i.

L'algorithme LLL [52] est l'un des plus important en théorie des nombres, avec de nombreuses applications en mathématique et en informatique. Il a été développé en 1982 par Lenstra, Lenstra and Lovász, avec une application à la factorisation des polynômes à coefficients rationnels. L'algorithme LLL permet de trouver un vecteur court dans un réseau :

Théorème 3 (LLL). *Soit L un réseau engendré par $(u_1, \ldots, u_n)$. L'algorithme LLL, étant donné $(u_1, \ldots, u_n)$, trouve en temps polynomial un vecteur b_1 tel que :*

$$\|b_1\| \leq 2^{(n-1)/4} \det(L)^{1/n}$$

En fait, l'algorithme LLL ne se content pas de calculer un vecteur court, il calcule une base *LLL-réduite* du réseau, définie de la manière suivante. Soient $b_1, \ldots, b_n$ des vecteurs linéairement indépendants dans $\mathbb{Z}^n$. D'après le procédé d'orthogonalisation de Gram-Schmidt, les vecteurs b_i^* $(1 \leq i \leq n)$ et les réels μ_{ij} $(1 \leq j < i \leq n)$ sont définis récursivement de la façon suivante :

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^* \tag{3}$$

$$\mu_{ij} = \frac{< b_i, b_j^* >}{< b_j^*, b_j^* >} \tag{4}$$

On obtient que $b_1^*, \ldots, b_n^*$ est une base orthogonale de $\mathbb{R}^n$.

Définition 1. *Une base $b_1, \ldots, b_n \in \mathbb{Z}^n$ est dite LLL-réduite si:*

$$|\mu_{ij}| \leq \frac{1}{2} \quad \textit{pour } 1 \leq j < i \leq n$$

et

$$|b_i^* + \mu_{i,i-1} b_{i-1}^*|^2 \geq \frac{3}{4} |b_{i-1}^*|^2 \quad \textit{pour } 1 < i \leq n$$

Proposition 1. *Soit $b_1, \ldots, b_n$ une base LLL-réduite pour un réseau de $\mathbb{Z}^n$, et soit $b_1^*, \ldots, b_n^*$ défini comme précédemment; on a alors :*

$$\|b_j\|^2 \leq 2^{i-1} \cdot \|b_i^*\|^2 \quad \textit{pour } 1 \leq j \leq i \leq n$$
$$\det L \leq \prod_{i=1}^{n} \|b_i\| \leq 2^{n(n-1)/4} \cdot \det L$$
$$\|b_1\| \leq 2^{(n-1)/4} \det(L)^{1/n}$$

La proposition précédente donne une borne sur la taille du premier vecteur b_1. Il est possible d'obtenir des bornes analogues sur la taille des vecteurs successifs de la base LLL-réduite.

3 Recherche de petites racines de polynômes modulaires à une variable

3.1 Le théorème de Coppersmith

Le théorème de Coppersmith pour la recherche de petites racines de polynômes modulaires à une variable est le suivant :

Théorème 4 (Coppersmith). *Soit un polynôme $P(x)$ de degré δ, modulo un entier N de factorisation inconnue. Il existe un algorithme permettant de trouver tous les entiers x_0 tels que $P(x_0) = 0 \bmod N$ et $|x_0| \leq N^{1/\delta}$, en temps polynomial en $(\log N, \delta)$.*

Nous rappelons la preuve du théorème de Coppersmith au prochain paragraphe, en suivant l'approche de Howgrave-Graham [49]. Le théorème de Coppersmith connaît de nombreuses applications en cryptographie. En particulier, il permet la cryptanalyse de RSA avec petit exposant public dans le cas suivant : on suppose que pour chiffrer un message m, on commence par lui concaténer à gauche une valeur constante et connue; le chiffré est alors donné par :

$$c = (B + m)^e \bmod N$$

où B est connu de l'attaquant. A partir de B et c, on peut donc définir le polynôme $p(x) = (B + x)^e - c$, et on a $p(m) = 0 \bmod N$. Si $m \leq N^{1/e}$, c'est à dire si le message m est suffisamment court, on peut donc retrouver m à partir de c en temps polynomial, sans connaître la factorisation de N. Ce résultat ne s'applique que pour un exposant e très petit (par exemple $e = 3$), et montre que l'encodage d'un message avec une bloc fixe n'est pas sûr. Au paragraphe 4, nous rappelons d'autres applications du théorème de Coppersmith.

3.2 Preuve du théorème de Coppersmith

On considère une liste de $r = \ell \cdot \delta + 1$ polynômes de la forme :

$$q_{ik}(x) = x^i \cdot N^{\ell-k} p^k(x) \bmod N^\ell$$

pour un certain paramètre $\ell \geq 1$, et en prenant $0 \leq i < \delta$ pour $0 \leq k < \ell$, et $i = 0$ pour $k = \ell$. On observe que pour tout i, k, on a $p(x_0) = 0 \bmod N^\ell$.

On considère maintenant une combinaison linéaire à coefficient entier $h(x)$ des polynômes $q_{ik}(x)$, telle que $h \neq 0$. On a alors également $h(x_0) = 0 \bmod N^\ell$. Le lemme suivant montre que si les coefficients du polynôme h sont suffisamment petit, alors l'égalité $h(x_0) = 0 \bmod N^\ell$ est aussi valable dans $\mathbb{Z}$, c'est à dire $h(x_0) = 0$. Il est alors possible de retrouver efficacement x_0 en utilisant un algorithme de recherche de racine de polynômes.

Lemme 1 (Howgrave-Graham). *Soit $h(x) \in \mathbb{Z}[x]$ égal à la somme d'au plus ω monômes. On suppose que $h(x_0) = 0 \bmod N^\ell$ où $|x_0| \leq X$ et $\|h(xX)\| < N^\ell/\sqrt{\omega}$. Alors l'égalité $h(x_0) = 0$ est aussi valable sur les entiers.*

Preuve. On a :

$$\begin{aligned} |h(x_0)| &= \left|\sum h_i x_0^i\right| = \left|\sum h_i X^i \left(\frac{x_0}{X}\right)^i\right| \\ &\leq \sum \left|h_i X^i \left(\frac{x_0}{X}\right)^i\right| \leq \sum |h_i X^i| \\ &\leq \sqrt{\omega}\|h(xX)\| < N^\ell \end{aligned}$$

Comme $h(x_0) = 0 \bmod N^\ell$, cela donne $h(x_0) = 0$. □

Le lemme précédent suggère de rechercher un vecteur court dans le réseau engendré par les polynômes $q_{ik}(xX)$. Plus précisément, on considère la matrice M de taille $r \times r$ dont la u-ème ligne est le vecteur formé par les coefficients du polynôme $q_{ik}(xX)$, où $k = \lfloor (u-1)/\delta \rfloor$ et $i = (u-1) - \delta k$. On donne une illustration de la matrice pour le cas $p(x) = x^2 + ax + b$ et $\ell = 1$:

$$M = \begin{bmatrix} X^2 & aX & b \\ 0 & NX & 0 \\ 0 & 0 & N \end{bmatrix}$$

On peut vérifier que dans le cas général, la matrice M est triangulaire supérieure, et que son déterminant est donné par :

$$\det L = X^{r(r-1)/2} \cdot N^{(r+1)\cdot\ell/2}$$

En utilisant LLL (théorème 3), on obtient donc un vecteur court qui correspond à un polynôme de la forme $h(xX)$, dont la norme est telle que :

$$\|h(xX)\| \leq 2^{(r-1)/4} (\det M)^{1/r} = 2^{(r-1)/4} X^{(r-1)/2} N^{(1+1/r)\ell/2}$$

D'après le lemme 1, la condition requise est $\|h(xX)\| < N^\ell/\sqrt{r}$, ce qui donne la condition suffisante suivante :

$$2^{(r-1)/4} X^{(r-1)/2} N^{(1+1/r)\ell/2} < \frac{N^\ell}{\sqrt{r}}$$

ce qui donne :

$$X < \frac{N^{\ell/r}}{\sqrt{2}} \cdot r^{-1/(r-1)}$$

On prend $\ell = \lceil \log N \rceil$. La condition suffisante devient alors :

$$X < \frac{1}{4} N^{1/\delta}$$

Comme l'algorithme LLL est polynomial en la taille des coefficients des vecteurs et en la dimension du réseau, le temps d'exécution est polynomial en $(\log N, \delta)$. Si on prend maintenant la condition suivante sur X, plus faible :

$$X < N^{1/\delta}$$

il suffit alors de diviser l'intervalle $[0, X]$ en quatre intervalles $[iX/4, (i+1)X/4]$ pour $i = 0, 1, 2, 3$, et d'appliquer l'algorithme précédent pour ces quatre intervalles. Ceci termine la preuve du théorème 4.

4 Applications du théorème de Coppersmith

4.1 L'attaque de Coppersmith sur les redondances courtes.

Nous avons rappelé au paragraphe 3 l'attaque sur le chiffrement RSA avec e petit lorsqu'une partie du message est connue. Cette attaque peut être étendue au cas suivant [17] : on suppose qu'avant de chiffrer un message on lui concatène une chaîne de bits aléatoires, de sorte que le chiffré peut s'écrire :

$$c = (m + r)^e \bmod N$$

avec r un entier aléatoire. Supposons qu'un attaquant obtienne une deuxième fois le chiffrement de m, avec un entier aléatoire $r' \neq r$:

$$c' = (m + r')^e \bmod N$$

La méthode suivante permet de retrouver m à partir de c, c', à condition que la taille de r et r' soit inférieure à $1/e^2$ fois la taille de N.

On définit les deux polynômes:

$$g(x, y) = x^e - c \quad \text{et} \quad g'(x, y) = (x + y)^e - c'$$

On voit que les deux polynômes ont pour racine commune $(m + r, r' - r)$ modulo N. L'entier $\Delta = r' - r$ est donc une racine du résultant $h(y) = \mathrm{res}_x(g, g')$ des polynômes g et g'. Le degré du polynôme h est au plus e^2. Donc si $r, r' < N^{1/e^2}$, on a $|\Delta| < N^{1/e^2}$, et Δ est une petite racine du polynôme h modulo N, que l'on peut retrouver efficacement en utilisant le théorème de Coppersmith. Lorsque Δ est connu, l'attaque de Franklin-Reiter décrite au chapitre 1 permet de retrouver m.

4.2 Attaque de RSA avec petit exposant d

Nous avons rappelé au chapitre précédent l'attaque de Wiener [75] permettant de retrouver la factorisation de $N = p \cdot q$ étant donné (N, e) si $d \leq N^{1/4}$. Cette borne a été améliorée en 1999 par Boneh et Durfee [10] pour un exposant $d < N^{0.292}$, en utilisant une variante du théorème de Coppersmith. La méthode consiste à considérer l'équation :

$$e \cdot d + k \cdot (N - x) = 1 \tag{5}$$

où e et N sont connus et d, k, et $x = p + q - 1$ sont inconnus, et à réécrire l'équation (5) sous la forme :

$$k \cdot (N - x) = 1 \bmod e \tag{6}$$

On obtient ainsi un polynôme à deux variables modulo e dont on cherche les petites racines k et x. Comme on est dans le cas d'un polynôme modulaire à deux variables, l'attaque est heuristique, mais on constate dans [10] qu'elle fonctionne toujours dans la pratique. Un problème ouvert intéressant consiste à essayer d'étendre la borne sur d jusqu'à $d < N^{0.5}$, borne pour laquelle il existe (heuristiquement) une unique solution à l'équation (6).

4.3 Factorisation de $N = p \cdot q$ lorsqu'une partie de p est connue

Supposons que l'on connaisse la moitié des bits des poids forts de p. Par division, on peut aussi retrouver la moitié des bits de poids fort de q. On écrit donc $p = P + x_0$ et $q = Q + y_0$ avec P et Q connus. La paire (x_0, y_0) est alors une petite racine du polynôme:

$$(P + x) \cdot (Q + y) - N = 0$$

On peut donc appliquer le théorème de Coppersmith pour les polynômes bivariés sur les entiers (théorème 2), et on montre que la connaissance de la moitié des bits de poids forts de p permet de retrouver x_0, y_0 et donc p et q. La même technique s'applique lorsqu'on connaît la moitié des bits de poids faible de p [18]. L'inconvénient de cette méthode est qu'elle est basée sur le deuxième théorème de Coppersmith (celui pour les polynômes à deux variables sur les entiers) dont l'algorithme est beaucoup plus difficile à mettre en oeuvre que pour le cas à une seule variable.

L'attaque a été simplifiée par Howgrave-Graham [49] qui a montré que l'on pouvait se ramener au cas d'un polynôme modulaire à une seule variable, en procédant de la façon suivante. On écrit toujours $p = P + x_0$ et l'on a :

$$N = 0 \bmod (P + x_0)$$

ce qui conduit à considérer les polynômes :

$$g_{i,k}(x) = x^i \cdot N^{m-k} \cdot (P + x)^k$$

pour un certain paramètre m et pour des entiers i, k choisis en fonction de m. On constate que $g_{i,k}(x_0) = 0 \bmod p^m$ pour tout i, k. Comme dans l'algorithme de Coppersmith classique, on va chercher une combinaison linéaire petite $h(x)$ des polynômes $g_{i,k}(x)$, ce qui permettra en utilisant une nouvelle fois le lemme d'Howgrave-Graham d'obtenir $h(x_0) = 0$ dans $\mathbb{Z}$ et donc de retrouver x_0. La seule différence est que l'on ne connaît pas le module p^m, mais cela n'est pas nécessaire pour appliquer la technique de Coppersmith.

4.4 Factorisation de $N = p^r q$ pour r grand

D. Boneh, G. Durfee, et N. Howgrave-Graham ont montré dans [11] que l'on pouvait factoriser les entiers de la forme $N = p^r q$ en temps polynomial lorsque $r \simeq \log p$. On peut voir cette attaque comme une extension de la précédente (pour laquelle $r = 1$). L'attaque est cependant théorique car dans la pratique, pour des tailles de N raisonnables, il est toujours plus efficace de factoriser N avec la méthode des courbes elliptiques de Lenstra [53] (de complexité sub-exponentielle).

Soit $N = p^r q$; la technique développée dans [11] est la suivante. On suppose tout d'abord que l'on connaît les ℓ bits de poids fort de p; on écrit $p = P + x_0$ avec P connu. De manière analogue à l'attaque précédente, on écrit :

$$N = 0 \bmod (P + x_0)^r$$

ce qui conduit à considérer cette fois-ci les polynômes :

$$g_{i,k}(x) = x^i \cdot N^{m-k} \cdot (P + x)^{rk}$$

pour un certain paramètre m et pour i, k choisis en fonction de m. On constate que $g_{i,k}(x_0) = 0 \bmod p^{rm}$ pour tout i, k. Comme précédemment, on cherche une combinaison linéaire petite $h(x)$ des polynômes $g_{i,k}(x)$, ce qui permet d'obtenir $h(x_0) = 0$ dans $\mathbb{Z}$ et donc de retrouver x_0. Les auteurs de [11] montrent que lorsque $r \simeq \log p$, les ℓ bits de poids fort de p nécessaires à l'attaque peuvent en fait être recherchés de manière exhaustive en temps polynomial (en $\log N$), et donc l'attaque est finalement polynomiale en $\log N$.

5 Équivalence déterministe entre factoriser N et calculer d

Il est bien connu qu'il existe un équivalence *probabiliste* entre calculer d et factoriser N. La preuve de cette équivalence est donnée dans l'article RSA d'origine [69]; elle est basée sur des résultats dus à Miller [57]. Allemande May a montré à la conférence Crypto 2004 que l'équivalence pouvait être rendu *déterministe*, en procédant de la façon suivante [55]. On connaît (N, e, d) et l'on doit obtenir la factorisation de N. Pour cela, on écrit :

$$e \cdot d + k \cdot (N - x_0) = 1$$

avec $x_0 = p + q - 1$. On obtient que (k, x_0) est une petite racine d'un polynôme à deux variables sur les entiers; on peut donc appliquer le théorème de Coppersmith pour les polynômes bivariés sur les entiers (théorème 2) et l'on obtient que si $e \cdot d \leq N^2$, alors on peut retrouver x_0 et donc p et q en temps polynomial.

Cette méthode a ensuite été simplifiée et rendue plus efficace par J.S. Coron et A. May dans [29], dont l'article reproduit en annexe sera publié au Journal of Cryptology. La technique consiste à se ramener au cas d'un polynôme modulaire à une seule variable en écrivant :

$$U = 0 \bmod (N - x_0)$$

où $U = e \cdot d - 1$ est connu, ce qui conduit à considérer les polynômes :

$$g_{ik}(x) = x^i \cdot (N - x)^k \cdot U^{m-k}$$

On constate que $g_{i,k}(x_0) = 0 \bmod (N - x_0)^m$ pour tout i, k. Une nouvelle fois, on va chercher une combinaison linéaire petite $h(x)$ des polynômes $g_{i,k}(x)$, ce qui permet d'obtenir $h(x_0) = 0$ dans $\mathbb{Z}$ et donc de retrouver x_0 et ainsi la factorisation de N.

6 Nouvel algorithme pour les polynômes bivariés sur les entiers

Dans ce paragraphe, nous présentons une variante plus simple de l'algorithme de Coppersmith pour trouver les petites racines d'un polynôme à deux variables sur les entiers. Cette variante a été publiée à la conférence Eurocrypt 2004 [23], et l'article est reproduit en annexe. La simplification obtenue est analogue à celle développée par Howgrave-Graham pour le cas des polynômes modulaires à une variable. L'avantage de cette nouvelle approche est que le réseau obtenu est maintenant de rang plein contrairement au réseau obtenu dans [18] qui n'est pas de rang plein et dont le déterminant est par conséquent plus difficile à calculer. Dans cette nouvelle approche, le réseau a une base triangulaire naturelle; on obtient donc immédiatement son déterminant et ensuite les bornes sur les racines. L'inconvénient de ce nouvel algorithme est qu'il est asymptotiquement moins efficace que l'algorithme de Coppersmith: en effet, notre algorithme a une complexité polynomiale si $XY < W^{2/3\delta-\varepsilon}$ pour tout $\varepsilon > 0$ fixé, alors que l'algorithme de Coppersmith requiert seulement $XY < W^{2/3\delta}$. On démontre donc le théorème suivant, plus faible que le théorème de Coppersmith (théorème 2) :

Théorème 5. *Soit $p(x, y)$ un polynôme irréductible à deux variables à coefficients entiers, de degré maximum δ séparément en chaque variable. Soient X et Y deux bornes supérieures sur les solution (x_0, y_0), et soit $W = \max_{i,j} |p_{ij}| X^i Y^j$. Soit $\varepsilon > 0$. Alors si $XY < W^{2/(3\delta)-\varepsilon}$, alors en temps polynomial en $(\log W, 2^\delta)$, on peut trouver toutes les paires d'entiers (x_0, y_0) telles que $p(x_0, y_0) = 0$, $|x_0| \leq X$ et $|y_0| \leq Y$.*

La preuve est donnée dans l'article reproduit en annexe. Dans la suite du paragraphe, nous illustrons la technique pour un polynôme bivarié de la forme :

$$p(x, y) = a + bx + cy + dxy$$

dont on cherche une petite racine (x_0, y_0) dans $\mathbb{Z}$. On suppose que p est un polynôme irréductible. Soit $W = \max\{|a|, |b|X, |c|Y, |d|XY\}$. On génère un entier n tel que $W \leq n < 2 \cdot W$ et $\mathrm{PGCD}(n, a) = 1$. On considère alors les polynômes :

$$\begin{aligned} q_{00}(x, y) &= a^{-1} p(x, y) \bmod n \\ &= 1 + b'x + c'y + d'xy \\ q_{10}(x, y) &= n \cdot x \\ q_{01}(x, y) &= n \cdot y \\ q_{11}(x, y) &= n \cdot x \cdot y \end{aligned}$$

On voit que l'on a $q_{ij}(x_0, y_0) = 0 \bmod n$. On recherche à l'aide de l'algorithme LLL une petite combinaison linéaire $h(x, y)$ de ces polynômes, en considérant le réseau :

$$L = \begin{bmatrix} 1 & b'X & c'Y & d'XY \\ 0 & nX & 0 & 0 \\ 0 & 0 & nY & 0 \\ 0 & 0 & 0 & nXY \end{bmatrix}$$

de déterminant $\det L = n^3(XY)^2$. A l'aide du lemme d'Howgrave-Graham, on obtient que si les coefficients de $h(x,y)$ sont suffisamment petits, alors l'équation $h(x_0,y_0) = 0 \bmod n$ est aussi valable dans $\mathbb{Z}$, et donc $h(x_0,y_0) = 0$. On montre aussi que si les coefficients de h sont suffisamment petits, alors le polynôme h ne peut pas être un multiple de p. Les polynômes p et h sont donc algébriquement indépendants et ont (x_0,y_0) pour racine commune. En prenant le résultant $Q(x) = \text{Resultant}_y(h(x,y),p(x,y))$, on obtient donc un polynôme non nul tel que $Q(x_0) = 0$, ce qui permet de retrouver x_0, et finalement y_0 en résolvant $p(x_0,y) = 0$.

Dans ce cas particulier, on obtient la borne $XY < W^{1/2}/16$ sur la taille des racines. Dans l'article reproduit en annexe, on montre qu'en considérant un ensemble plus grand de polynômes $q_{ij}(x,y)$, on retrouve la borne $XY < W^{2/3-\varepsilon}$ du théorème 5.

Conclusion

Nous avons vu dans ce mémoire que l'approche correcte en cryptographie consiste à d'abord formaliser la notion de sécurité que l'on veut atteindre, pour ensuite construire un schéma qui satisfait cette notion de sécurité sous une hypothèse de complexité bien définie : c'est l'approche de la sécurité prouvée, qui est maintenant dominante en cryptographie. Nous avons rappelé comment se formalisent la sécurité d'un schéma de chiffrement à clef publique et la sécurité d'un schéma de signature. Nous avons proposé une preuve améliorée (et optimale) pour le schéma de signature PSS. Nous avons aussi proposé la première preuve de sécurité pour les standards de signature ISO 9796-2 et PKCS #1 v1.5. Finalement, nous avons proposé une nouvelle définition de sécurité pour les fonctions de hachage, plus forte que la résistance aux collisions, et proposé plusieurs constructions pratiques qui permettent de satisfaire cette définition.

Dans ce mémoire, nous avons aussi rappelé les principales attaques contre certaines variantes du schéma RSA, en particulier les attaques basées sur le théorème de Coppersmith. Ces attaques ne mettent pas en cause l'algorithme RSA lui-même, mais plutôt son utilisation incorrecte. Nous avons aussi proposé un algorithme plus simple que celui de Coppersmith pour trouver les petites racines d'un polynôme à deux variables sur les entiers.

Bibliographie

1. FIPS 46. Data encryption standard. *Federal Processing Standards Publication 46, U.S. Department of Commerce*, 1977.
2. ISO/IEC 9796. Information technology - security techniques - digital signature scheme giving message recovery, part 1 : Mechanisms using redundancy, 1999.
3. ISO/IEC 9796-2. Information technology - security techniques - digital signature scheme giving message recovery, part 2 : Mechanisms using a hash-function, 1997.
4. N. Baric et B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233, pages 480–494. Springer-Verlag, 1997. Lecture Notes in Computer Science.
5. M. Bellare, A. Desai, D. Pointcheval, et P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO' 98*, volume 1462, pages 26–45. Springer-Verlag, 1998. Lecture Notes in Computer Science.
6. M. Bellare et P. Rogaway. Random oracles are practical : a paradigm for designing efficient protocols. In *Proceedings of the first annual conference on computer and communication security*, 1993.
7. M. Bellare et P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology - EUROCRYPT '94*, volume 950, pages 92–111. Springer-Verlag, 1995. Lecture Notes in Computer Science.
8. M. Bellare et P. Rogaway. The exact security of digital signatures - how to sign with RSA and Rabin. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070, pages 399–416. Springer-Verlag, 1996. Lecture Notes in Computer Science.
9. D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society (AMS)*, 46(2):203–213, 1999.
10. D. Boneh et G. Durfee. Cryptanalysis of RSA with private key d less than $n^{0.292}$. In *Advances in Cryptology - EUROCRYPT '99*, volume 1592. Springer-Verlag, 1999. Lecture Notes in Computer Science.
11. D. Boneh, G. Durfee, et N. Howgrave-Graham. Factoring n = $p^r q$ for large r. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 326–337. Springer-Verlag, 1999.
12. D. Boneh et R. Venkatesan. Breaking RSA may not be equivalent to factoring. In *EUROCRYPT' 98*, pages 59–71, 1998.
13. E. Brier, C. Clavier, J.S. Coron, et D. Naccache. Cryptanalysis of RSA signatures with fixed-pattern padding. In *Proceedings of CRYPTO 2001*, volume 2139. Springer-Verlag, 2001. Lecture Notes in Computer Science.
14. R. Canetti, O. Goldreich, et S. Halevi. The random oracle methodology, revisited. *STOC' 98, ACM*, 1998.

15. D. Coppersmith. Finding small solutions to small degree polynomials. In *Proc. of CALC '01*. Springer-Verlag. LNCS.
16. D. Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070. Springer-Verlag, 1996. Lecture Notes in Computer Science.
17. D. Coppersmith. Finding a small root of a univariate modular equation. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070, pages 155–165. Springer-Verlag, 1996. Lecture Notes in Computer Science.
18. D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *J. of Cryptology*, 10:233–260, 1997.
19. D. Coppersmith, J.S. Coron, F. Grieu, S. Halevi, C. Jutla, D. Naccache, et J.P. Stern. Cryptanalysis of ISO/IEC 9796-1. *Submitted to the Journal of Cryptology.*
20. D. Coppersmith, M.K. Franklin, J. Patarin, et M.K. Reiter. Low-exponent RSA with related messages. In Ueli Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070, pages 1–9. Springer-Verlag, 1996. Lecture Notes in Computer Science.
21. D. Coppersmith, S. Halevi, et C. Jutla. ISO 9796-1 and the new forgery strategy. Technical report, Contribution de recherche au groupe P1363, 1999.
22. D. Coppersmith et A. Shamir. Lattice attacks on NTRU. In *Proc. of Eurocrypt '97*. LNCS 1233, Springver-Verlag, 1997.
23. J.S. Coron. Finding small roots of bivariate integer polynomial equations revisited. In *Proceedings of EUROCRYPT 2004*, volume 3027. Springer-Verlag. Lecture Notes in Computer Science.
24. J.S. Coron. Optimal security proofs for PSS and other signature schemes. In *EUROCRYPT 2002*, pages 272–287, 2002.
25. J.S. Coron. Security proof for partial-domain hash signature schemes. In *CRYPTO 2002*, pages 613–626, 2002.
26. J.S. Coron, Y. Desmedt, D. Naccache, A. Odlyzko, et J.P. Stern. Index calculation attacks on RSA signature and encryption. *To appear in Design, Codes and Cryptograpy (DCC).*
27. J.S. Coron, Y. Dodis, C. Malinaud, et P. Puniya. Merkle-damgård revisited : how to construct a hash function. In *Proceedings of CRYPTO 2005.*
28. J.S. Coron, M. Joye, D. Naccache, et P. Paillier. Universal padding schemes for RSA. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 226–241. Springer-Verlag, 2002.
29. J.S. Coron et A. May. Deterministic polynomial time equivalence of computing the RSA secret key and factoring. *To appear in Journal of Cryptology.*
30. J.S. Coron, D. Naccache, et J.P. Stern. On the security of RSA padding. In *Advances in Cryptology - CRYPTO '99*, volume 1666, pages 1–18. Springer-Verlag, 1999. Lecture Notes in Computer Science.
31. R. Cramer et V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO'98 Proceedings, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk, ed., Springer-Verlag, 1998.*
32. R. Cramer et V. Shoup. Signature schemes based on the strong RSA assumption. In *Proc. 6th ACM Conf. on Computer and Communications Security*, 1996. Disponible à `http://www.shoup.net/`.

33. J. Daemen et V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer-Verlag, 2002.
34. W. de Jonge et D. Chaum. Attacks on some RSA signatures. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85*, volume 218, pages 18–27. Springer-Verlag, 1986. Lecture Notes in Computer Science.
35. Y. Desmedt et A.M. Odlyzko. A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85*, volume 218, pages 516–522. Springer-Verlag, 1986. Lecture Notes in Computer Science.
36. W. Diffie et M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
37. D. Dolev, C. Dwork, et M. Naor. Non-malleable cryptography. *In Proc. of the 23rd Symposium on the theory of Computing, ACM*, 1991.
38. E. Fujisaki, T. Okamoto, D. Pointcheval, et J. Stern. RSA–OAEP is secure under the RSA assumption. In J. Kilian, editor, *Advances in Cryptology — Proceedings of CRYPTO '2001 (19 – 23 august 2001, Santa Barbara, California, USA)*, volume 2139 of *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
39. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, 1985.
40. R. Gennaro, S. Halevi, et T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Proceedings of EUROCRYPT '99*, volume 1592, pages 123–139. Springer-Verlag, 1999. Lecture Notes in Computer Science.
41. M. Girault et J.-F. Misarsky. Selective forgery of RSA signatures using redundancy. In Walter Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233, pages 495–507. Springer-Verlag, 1997. Lecture Notes in Computer Science.
42. S. Goldwasser et S. Micali. Probabilistic encryption. *J. of Computer and System Sciences*, 28:270–299, 1984.
43. S. Goldwasser, S. Micali, et R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. of computing*, 17(2):281–308, april 1988.
44. S. Goldwasser, S. Micali, et P. Tong. Why and how to establish a private code on a public network. *Proc. 23rd IEEE Symp. on Foundations of Comp. Science*, pages 134–144, 1982.
45. S. Goldwasser et Y. Tauman. On the (in)security of the Fiat-Shamir paradigm. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (2003)*.
46. H. Handschuh et D. Naccache. Shacal. In *B. Preneel, Ed., First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000.*
47. K. Hickman. The SSL protocol, December 1995. Article disponible à l'adresse : `http://www.netscape.com/newsref/std/ssl.html`.
48. J. Hoffstein, J. Pipher, et J. H. Silverman. NTRU: A ring-based public key cryptosystem. *Lecture Notes in Computer Science*, 1423, 1998.
49. N. A. Howgrave-Graham. Finding small roots of univaraite modular equations revisited. In *Cryptography and Coding*, volume 1355. Springer-Verlag. LNCS.
50. B. Kalisky et M. Robshaw. The secure use of RSA. *CryptoBytes*, 1(3):7–13, 1995.

51. A. Lenstra et H. Lenstra. The development of the number field sieve. In *Lecture Notes in Mathematics*, volume 1554. Springer-Verlag, 1993.
52. A.K. Lenstra, H.W. Lenstra Jr., et L. Lovász. Factoring polynomials with rational coefficients. *Mathematischen Annalen*, 261:515–535, 1982.
53. H. Lenstra. Factoring integers with elliptic curves. *Annals of mathematics*, 126, 1987.
54. A. Boldyreva M. Bellare et A. Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In *Proccedings of Eurocrypt 2004*.
55. A. May. Computing the rsa secret key is deterministic polynomial time equivalent to factoring. In *Proc. of CRYPTO '04*. Springer-Verlag. LNCS.
56. A. Menezes, P. van Oorschot, et S. Vanstone. *Handbook of Applied Cryptography*. 1996.
57. G. L. Miller. Riemann's hypothesis and tests for primality. In *Seventh Annual ACM Symposium on the Theory of Computing*, pages 234–239, 1975.
58. J.-F. Misarsky. A multiplicative attack using LLL algorithm on RSA signatures with redundancy. In Burt Kaliski, editor, *Advances in Cryptology - CRYPTO '97*, volume 1294, pages 221–234. Springer-Verlag, 1997. Lecture Notes in Computer Science.
59. M. Naor et M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *ACM Symposium on Theory of Computing*, pages 427–437, 1990.
60. P. Q. Nguyen et J. Stern. The two faces of lattices in cryptology. *Lecture Notes in Computer Science*, 2146, 2001.
61. J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology - Crypto 2002 Proceedings*.
62. A. Odlyzko. The rise and fall of knapsack cryptosystems. In *Cryptology and Computational Number Theory, Proceedings of Symposia in Applied Mathematics, Vol. 42, pp.75-88, 1990.*
63. National Institute of Standards et Technology. Advanced encryption standard. *FIPS-197, NIST, Nov. 2001. Available at http://csrc.nist.gov/encryption/.*
64. National Institute of Standards et Technology. Secure hash standard. FIPS publication 180-1, April 1994.
65. T. Okamoto et D. Pointcheval. REACT: Rapid enhanced-security asymmetric cryptosystem transform. In *CT-RSA*, Lecture Notes in Computer Science, pages 159–175. Springer, 2001.
66. T. Okamoto et A. Shiraishi. A fast signature scheme based on quadratic inequalities. *Proc. of the 1985 Symposium on Security and Privacy*, April 1985.
67. D. Pointcheval et J. Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070, pages 387–398. Springer-Verlag, 1996. Lecture Notes in Computer Science.
68. Charles Rackoff et Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 433–444. Springer-Verlag, 1992.
69. R. Rivest, A. Shamir, et L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21-2:120–126, 1978.
70. PKCS #1 v2.1: *RSA cryptography specifications*. Document disponible à `www.rsasecurity.com/rsalabs/pkcs`.

71. B. Schneier. *Cryptographie appliquée*. International Thomson Publishing France, Paris, 1995.
72. RSA Data Security. PKCS #1: *RSA Encryption Standard*, November 1993. Version 1.5.
73. Victor Shoup. OAEP reconsidered. In *Proceedings of CRYPTO 2001*, pages 239–259, 2001.
74. D. Stinson. *Cryptographie : théorie et pratique*. International Thomson Publishing France, 1996. Traduction par S. Vaudenay de "Cryptography: theory and practice", CRC Press, Inc., 1995.
75. M. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Informations Theory*, 36:553–558, 1990.

Partie II

Curriculum Vitae et Publications

Curriculum Vitae

1 État Civil

Jean-Sébastien Coron, né le 28 juin 1975 à Bordeaux, marié, deux enfants.

2 Formation

– 1998 - 2001 : **École Polytechnique, Palaiseau.**
Thèse de Doctorat en cryptographie, soutenue le 16 mai 2001: "Cryptanalyses et preuves de sécurité de schémas à clef publique", dirigée par Jacques Stern. Membres du jury: M. Bellare, M. Girault, D. Naccache, R. Rivest, A. Shamir, J. Stern, J.M. Steyaert et S. Vaudenay.

– 1995 - 1998 : **Ecole Normale Supérieure, Paris.**
DEA Algorithmique, Paris VI, mention Bien.
Licence et Maîtrise de physique.

– 1993 - 1995 : **Lycée Thiers, Marseille.**
Classes préparatoires aux grandes écoles. Admis à l'École Polytechnique et à l'École Normale Supérieure.

3 Cursus Professionnel

– Depuis Octobre 2004 : **Université du Luxembourg**
Assistant-Professeur d'informatique.
- Cours de cryptographie et de programmation.
- Responsable du thème "Sécurité et Confiance" du Master of Computer Science de l'Université du Luxembourg.
- Recherche en cryptographie.

– 1998 - 2004 : **Gemplus**
Ingénieur de Recherche au département "Sécurité de l'information".
- Recherche en Cryptographie: conception, analyse, optimisation et implémentation de schémas cryptographiques. Cryptanalyses de schémas existants.
- Attaques physiques sur carte: analyse des attaques physiques contre la carte à puce, et conception de contre-mesures.

4 Projets de Recherche

– Ancien membre du réseau Européen d'excellence en cryptologie ECRYPT, et ancien responsable du groupe de travail "Cryptanalyse asymétrique" de ce réseau.

– Participation au projet Crypto++ du RNRT - Réseau National de Recherche en Télécommunications, "Protocoles cryptographiques pour la sécurité multi-acteurs".
– Participation au projet de recherche CRYNOSEC de l'Université du Luxembourg, "Etude de nouvelles primitives cryptographiques".

5 Conférences invitées

- *Cryptanalysis of ISO 9796-1 and ISO 9796-2 standards*, Royal Holloway, University of London. Séminaire de Cryptographie, Mars 1999.

- *Optimal security proofs for signature schemes*, Monte Verita Workshop, Switzerland, Mars 2001.

- *Optimal security proofs for PSS*, École Normale Supérieure de Paris, Séminaire de Cryptographie, Avril 2002.

- *Equivalence between the random oracle model and the random cipher model*, Dagstuhl Worshop, Germany, Septembre 2002.

- *Cryptanalysis of Augot and Finiasz Cryptosystem of Eurocrypt 2003*, École Normale Supérieure de Paris, Séminaire de Cryptographie, March 2003.

- *How to use RSA in practice ?*, Quo vadis cryptology international workshop, Warsaw, Poland, Mai 2003.

- *Finding small roots of bivariate integer polynomial equations revisited*, Royal Holloway, University of London. Séminaire de Cryptographie, Avril 2004.

6 Valorisation de la recherche

Membre du comité de programme des conférences suivantes:

– CRYPTO 2003, EUROCRYPT 2004, CRYPTO 2005, EUROCRYPT 2006.
– CHES 2001, CHES 2002, CHES 2003, CHES 2006.
– CT-RSA 2005, PKC 2006.

7 Encadrement de la recherche

– 2004: Stage de recherche - Cécile Malinaud, *Relations entre le modèle de l'oracle aléatoire et le modèle du block-cipher idéal*
– 2000: Stage de DEA - Christophe Tymen, *Etude théorique et pratique d'un schéma de signature efficace basé sur les courbes elliptiques.*
– 1999: Stage de recherche - Nora Dabbous, *Implémentation efficace du schéma de signature ECDSA sur carte à puce.*

8 Enseignement

– *Méthodologie de la programmation*, Université du Luxembourg, 2004.
– *Systèmes d'exploitation*, Université du Luxembourg, 2004.
– *Introduction à la cryptographie*, Université du Luxembourg, 2004.

Publications

1 Articles de journaux

1. J.S. Coron et A. May, *Deterministic Polynomial Time Equivalence of Computing the RSA Secret Key and Factoring*. A paraître dans Journal of Cryptology.
2. J.S. Coron, Y. Desmedt, D. Naccache, A. Odlyzko et J.P. Stern, *Index Calculation Attacks on RSA Signature and Encryption*. A paraître dans Designs, Codes and Cryptography.
3. J.S. Coron, D. Naccache et P. Kocher, *Statistics and secret leakage*. ACM Transactions on Embedded Computing Systems (TECS). Volume 3 , Issue 3 (August 2004), Pages: 492 - 508, 2004.

2 Articles dans conférences avec comité de lecture

4. J.S. Coron, D. Naccache and J.P. Stern. On the security of RSA padding. In *Advances in Cryptology - CRYPTO '99*, volume 1666, pages 1–18. Springer-Verlag, 1999. Lecture Notes in Computer Science.
5. J.S. Coron, H. Handschuh and D. Naccache. ECC: do we need to count ? In *Advances in Cryptology - ASIACRYPT '99*, volume 1716, pages 122–134. Springer-Verlag, 1999. Lecture Notes in Computer Science.
6. J.S. Coron, M. Joye, D. Naccache and P. Paillier. New attacks on PKCS#1 v1.5 encryption. In B. Preneel, editeur, *Prooceedings of EUROCRYPT 2000*, volume 1807, pages 369–381. Springer Verlag, 2000. Lecture Notes in Computer Science.
7. J.S. Coron and D. Naccache. Security analysis of the Gennaro-Halevi-Rabin signature scheme. In B. Preneel, editeur, *Prooceedings of EUROCRYPT 2000*, volume 1807, pages 91–101. Springer Verlag, 2000. Lecture Notes in Computer Science.
8. J.S. Coron. On the exact security of full domain hash. In M. Bellare, editeur, *Prooceedings of CRYPTO 2000*, volume 1880, pages 229–235. Springer Verlag, 2000. Lecture Notes in Computer Science.
9. J.S. Coron, F. Koeune and D. Naccache. From fixed-lenght to arbitrary-length padding schemes. In *Advances in Cryptology - ASIACRYPT 2000*, volume 1976, pages 90–96. Springer-Verlag, 2000. Lecture Notes in Computer Science.
10. J.S. Coron, E. Brier, C. Clavier, and D. Naccache. Cryptanalysis of RSA signatures with fixed-pattern padding. In *CRYPTO 2001*. Springer-Verlag, 2001. Lecture Notes in Computer Science.
11. J.S. Coron. Optimal Security Proofs for PSS and Other Signature Schemes. In *EUROCRYPT 2002*. Springer-Verlag, 2002. Lecture Notes in Computer Science.
12. J.S. Coron. Security Proof for Partial-Domain Hash Signature Schemes. In *CRYPTO 2002*. Springer-Verlag, 2002. Lecture Notes in Computer Science.
13. J.S. Coron, M. Joye, D. Naccache, P. Paillier Universal Padding Schemes for RSA. In *CRYPTO 2002*. Springer-Verlag, 2002. Lecture Notes in Computer Science.
14. J.S. Coron, D. Naccache. Boneh et al's k-Element Aggregate Extraction Assumption Is Equivalent to The Diffie-Hellman Assumption. In *ASIACRYPT 2003*. Springer-Verlag, 2003. Lecture Notes in Computer Science.

15. J.S. Coron. On finding small roots of bivariate polynomial equations revisited. In *EUROCRYPT 2004*. Springer-Verlag, 2004. Lecture Notes in Computer Science.
16. J.S. Coron, Y. Dodis, C. Malinaud and P. Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In *CRYPTO 2005*. Springer-Verlag, 2005. Lecture Notes in Computer Science.
17. J.S. Coron and D. Naccache. An accurate evaluation of maurer's universal test. In *Selected Areas in Cryptography, SAC '98*, volume 1556, pages 57–71. Springer-Verlag, 1998. Lecture Notes in Computer Science.
18. J.S. Coron and D. Naccache. On the security of RSA screening. In *Proceedings of PKC '99*, volume 1560, pages 197–203. Springer-Verlag, 1999. Lecture Notes in Computer Science.
19. J.S. Coron. On the security of random sources. In *Proceedings of PKC '99*, volume 1560, pages 29–42. Springer-Verlag, 1999. Lecture Notes in Computer Science.
20. J.S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Proceedings of CHES '99*, volume 1717, pages 292–302. Springer-Verlag, 1999. Lecture Notes in Computer Science.
21. C. Clavier, J.S. Coron and N. Dabbous. Differential power analysis in the presence of hardware countermeasures. In *CHES 2000*, volume 1965, pages 252–263. Springer-Verlag, 2000. Lecture Notes in Computer Science.
22. J.S. Coron and L. Goubin. On boolean and arithmetic masking against differential power analysis. In *CHES 2000*, volume 1965, pages 231–237. Springer-Verlag, 2000. Lecture Notes in Computer Science.
23. J.S. Coron, P. Kocher and D. Naccache. Statistics and secrand leakage. In *Financial Cryptography 2000*. Springer-Verlag, 2000. Lecture Notes in Computer Science.
24. J.S. Coron, D. M'Raïhi and C. Tymen. Fast Generation of Pairs $(k, [k]P)$ for Koblitz Elliptic Curves. In *SAC 2001*. Springer-Verlag, 2001. Lecture Notes in Computer Science.
25. J.S. Coron, H. Handschuh, M. Joye, P. Paillier, D. Pointcheval, C. Tymen. Optimal Chosen-Ciphertext Secure Encryption of Arbitrary-Length Messages. In *PKC 2002*. Springer-Verlag, 2002. Lecture Notes in Computer Science.
26. J.S. Coron, H. Handschuh, M. Joye, P. Paillier, D. Pointcheval, C. Tymen. GEM: A Generic Chosen-Ciphertext Secure Encryption Method. In *CT-RSA 2002*. Springer-Verlag, 2002. Lecture Notes in Computer Science.
27. J.S. Coron, A. Tchoulkine. A New Algorithm for Switching from Arithmetic to Boolean Masking In *CHES 2003*. Springer-Verlag, 2003. Lecture Notes in Computer Science.
28. J.S. Coron and D. Naccache. Cryptanalysis of a Zero-Knowledge Identification Protocol of Eurocrypt '95. In *CT-RSA 2004*. Springer-Verlag, 2004. Lecture Notes in Computer Science.
29. J.S. Coron. Cryptanalysis of a Public-key Encryption Scheme Based on the Polynomial Reconstruction Problem. In *PKC 2004*. Springer-Verlag, 2004. Lecture Notes in Computer Science.

3 Brevets

Voici les brevets déposés dans le cadre de mon travail à Gemplus. Ces brevets peuvent se diviser en deux catégories: les brevets cryptographiques, décrivant de nouvelles primitives

cryptographiques, et les brevets "side-channel", destinés à protéger la carte à puce contre ce type d'attaque.

3.1 Brevets Cryptographiques

1. J.S. Coron, D. Naccache, J. Stern, "Signature schemes based on discrete logarithm with partial or total message recovery". Patent Number: FR2797127. Publication date: 2001-02-02. International Publication Number: WO0110078.
2. J.S. Coron, C. Tymen, "Cryptography method on elliptic curves", Patent Number: FR2807898. Publication date: 2001-10-19. Patent Number: US2003152218. Publication Date: 2003-08-14. International Publication Number: WO0180481.
3. J.S. Coron, D. Naccache, "Method for encoding long messages for RSA electronic signature schemes", Patent Number: FR2814619, EP1325584. Publication date: 2002-03-29. Patent Number: US2003165238. Publication date: 2003-09-04. International Publication Number: WO0228010.
4. J.S. Coron, M. Joye and P. Paillier, " Method for enhancing security of public key encryption scheme". Patent Number: FR2818471. Publication date: 2002-06-21. Patent Number: US2004120519. Publication date: 2004-06-24 International Publication Number: WO0251065
5. J.S. Coron, "Method for determining the size of a random variable for an electronic signature scheme". Patent Number: FR2828353. Publication date: 2003-02-07. International Publication Number: WO03013053.
6. J.S. Coron, D. Naccache, "Method for accelerated transmission of electronic signature". Patent Number: US2002188850. Publication date: 2002-12-12. International Publication Number: WO0228011.
7. J.S. Coron, M. Joye, D. Naccache and P. Paillier, "Data encryption method, cryptographic system and associated component". Patent Number: WO2004012372. Publication date: 2004-01-30.
8. J.S. Coron and D. Naccache, "Method of reducing the size of an RSA or Rabin signature". Patent Number: FR2829333. Publication date: 2003-03-07. International Publication Number: WO03021864.

3.2 Brevets "Side-channel"

9. J.S. Coron, C. Clavier, "Countermeasure method in an electronic component using a secret key cryptographic algorithm". Patent Number: EP1125394. Publication date: 2000-05-05. International Publication Number: WO0027068.
10. J.S. Coron, O. Benoit, N. Feyt, D. Naccache, " Method for countermeasure in an electronic component using a secret key algorithm". Patent Number: EP1198921; Publication date: 2000-08-18. International Publication Number: WO0049765.
11. J.S. Coron, "Countermeasure method in an electric component implementing an elliptical curve type public key cryptography algorithm". Patent Number: FR2791496. Publication date: 2000-09-29. International Publication Number: WO0059157.

12. J.S. Coron, "Countermeasure procedures in an electronic component implementing an elliptical curve type public key encryption algorithm". Patent Number: FR2791497. Publication date: 2000-09-29. International Publication Number: WO0059156.
13. J.S. Coron, D. Naccache, "Method for improving a random number generator to make it more resistant against attacks by current measuring". Patent Number: FR2796477. Publication date: 2001-01-19. International Publication Number: WO0106350.
14. J.S. Coron, P. Paillier, "Countermeasure method in an electronic component which uses an RSA-type public key cryptographic algorithm". Patent Number: FR2799851. Publication date: 2001-04-20. International Publication Number: WO0128153.
15. J.S. Coron, " countermeasure methods in an electronic component using a Koblitz elliptic curve public key cryptographic algorithm", Patent Number: FR2810821. Publication date: 2001-12-28. International Publication Number: WO0201343.
16. J.S. Coron, "Counter-measure method in an electronic component using a secret key encryption algorithm" Patent Number: FR2818472, EP1348275. Publication date: 2002-06-21. International Publication Number: WO0251064.
17. J.S. Coron, "Countermeasure methods in an electronic component using an rsa-type public key encryption algorithm", Patent Number: FR2818473. Publication date: 2002-06-21. International Publication Number: WO0250658.
18. J.S. Coron, M. Joye and P. Paillier, "Execution of cryptographic algorithms based on the Chinese Remainder theorem, CRT, for use with chip cards, etc., which by changing the order of calculation runs faster than existing CRT algorithms". Patent Number: FR2819663, EP1352494. Publication date: 2002-07-19. International Publication Number: WO02058321.

Publication:	Nombre
Meilleures conférences (Eurocrypt, Crypto)	10
Autres conférences	16
Journaux	3
Brevets	18

Partie III

Articles Joints

Cryptanalyse

Nous avons classé les trois articles ci-dessous dans la catégorie "cryptanalyse", bien que seul le premier soit une cryptanalyse au sens strict. Le point commun de ces trois articles est qu'ils mettent en oeuvre une analyse mathématique pour résoudre un problème ayant des applications cryptographiques. Dans le premier, on utilise l'algèbre linéaire pour casser un schéma à clef publique proposé lors de la conférence Eurocrypt 2003. Dans les deuxièmes et troisièmes, on utilise la réduction de réseaux pour trouver les petites racines d'un polynômes à deux variables sur les entiers, et pour montrer l'équivalence déterministe entre connaître l'exposant privé d de RSA et factoriser N.

1. **"Cryptanalysis of a Public-key Encryption Scheme Based on the Polynomial Reconstruction Problem"**, Jean-Sébastien Coron, Proceedings de PKC 2004.

   Dans cet article, on montre que le schéma de chiffrement à clef publique proposé par Augot et Finiasz à la conférence Eurocrypt 2003 n'est pas sûr. On montre en effet comment retrouver le message clair à partir du chiffré et de la clef publique, en temps polynomial. L'attaque est une variante de l'algorithme de Berlekamp-Welsh, et ne met en oeuvre que de l'algèbre linéaire.

2. **"Finding Small Roots of Bivariate Integer Polynomial Equations Revisited"**, Jean-Sébastien Coron, Proceedings de Eurocrypt 2004.

   Dans cet article, on donne un algorithme plus simple que l'algorithme de Coppersmith pour trouver les petites racines d'un polynôme à deux variables sur les entiers. La technique est basée sur l'algorithme de réduction de réseaux LLL. Une application de cet algorithme est la factorisation en temps polynomial d'un module RSA $N = pq$ lorsque la moitié des bits de poids fort ou de poids faible de p est connue.

3. **"Deterministic Polynomial Time Equivalence of Computing the RSA Secret Key and Factoring"**, Jean-Sébastien Coron et Alexander May, à paraître dans Journal of Cryptology.

   Il est bien connu qu'il existe un algorithme *probabiliste* polynomial qui à partir de (N, e, d) renvoie les facteurs p et q de N. On donne dans cet article le premier algorithme *déterministe* permettant de résoudre le même problème. On utilise pour cela une variante du théorème de Coppersmith permettant de trouver les petites racines d'un polynôme modulaire à une variable.

Cryptanalysis of a Public-key Encryption Scheme Based on the Polynomial Reconstruction Problem

PKC 2004

Jean-Sébastien Coron

Gemplus Card International
34 rue Guynemer
Issy-les-Moulineaux, F-92447, France
jean-sebastien.coron@gemplus.com

Abstract. We describe a cryptanalysis of a public-key encryption scheme based on the polynomial reconstruction problem, published at Eurocrypt 2003 by Augot and Finiasz. Given the public-key and a ciphertext, we recover the corresponding plaintext in polynomial time. Our technique is a variant of the Berlekamp-Welsh algorithm. We also describe a cryptanalysis of the reparation published by the authors on the IACR eprint archive, using a variant of the previous attack. Both attacks are practical as given the public-key and a ciphertext, one recovers the plaintext in a few minutes on a single PC.

Key-Words: Cryptanalysis, Augot and Finiasz cryptosystem, Polynomial Reconstruction Problem, Reed-Solomon codes.

1 Introduction

We describe a cryptanalysis of a public-key encryption scheme recently proposed by Augot and Finiasz [1]. The scheme is based on the polynomial reconstruction (PR) problem [10], which is the following:

Problem 1 (Polynomial Reconstruction). Given n, k, ω and $(x_i, y_i)_{i=1\ldots n}$, output any polynomial p such that $\deg p < k$ and $p(x_i) = y_i$ for at least $n - \omega$ values of i.

This problem has an equivalent formulation in terms of the decoding of Reed-Solomon error-correcting codes [11]. The problem can be solved in polynomial time when the number of errors ω is such that $\omega \leq (n-k)/2$, using the Berlekamp-Welsh algorithm [3]. This has been improved to $\omega \leq n - \sqrt{kn}$ by Guruswami and Sudan [7].

When the number of errors is larger, no polynomial time algorithm is known for the PR problem. Therefore, some cryptosystem have been constructed based on the hardness of the PR problem; for example, an oblivious polynomial evaluation scheme [10], and a semantically secure symmetric cipher [8].

At Eurocrypt 2003, Augot and Finiasz proposed a new public-key encryption scheme based on the polynomial reconstruction problem [1]. A security level exponential in terms of the parameters was conjectured. However, we provide a complete cryptanalysis of the cryptosystem: given the public key pk and a ciphertext c, we recover the corresponding plaintext m in polynomial time. Therefore, the scheme is not one-way and cannot be used

in any application. Our technique is a variant of the Berlekamp-Welsh algorithm [3] for solving the PR problem.

After the publication of our attack in the IACR eprint archive [5], a reparation of the cryptosystem was published by Augot, Finiasz and Loidreau in [2]. The reparation is based on the trace operator, and is resistant against the previous attack. However, we describe a new cryptanalysis of the repaired scheme. Given the public-key and a ciphertext, we can still recover the corresponding plaintext in polynomial time. Our technique is again a variant of the Berlekamp-Welsh algorithm. Both attacks work very well in practice, as for the proposed parameters, one recovers the plaintext in a few minutes on a single PC.

2 Augot and Finiasz' Cryptosystem

In this section, we recall the original cryptosystem proposed by Augot and Finiasz at Eurocrypt 2003. As in [1], we first recall some basic definitions of Reed-Solomon codes.

2.1 Reed-Solomon Codes

Let F_q be the finite field with q elements and let $x_1, \cdots, x_n$ be n distinct elements of F_q. We denote by ev the following map:

$$ev : \begin{cases} F_q[X] \to F_q^n \\ p(X) \to (p(x_1), \ldots, p(x_n)) \end{cases}$$

Definition 1. *The Reed-Solomon code of dimension k and length n over F_q is the following set of n-tuples (codewords):*

$$RS_k = \{ev(f); f \in F_q[X], \deg f < k\}$$

where $F_q[X]$ is the set of univariate polynomials with coefficients in F_q.

The *weight* of a word $c \in F_q^n$ is the number of non-zero coordinates in c. The *Hamming distance* between two words x and y is the weight of $x - y$. Formally, the problem of decoding Reed-Solomon code is the following:

Problem 2 (Reed-Solomon decoding). Given a Reed-Solomon code RS_k of length n, ω an integer and a word $y \in F_q^n$, find any codeword in RS_k at distance less than ω of y.

The smallest weight of non-zero codewords in RS_k is $n - k + 1$. Therefore, when $\omega \leq (n-k)/2$, the solution to Reed-Solomon decoding is guaranteed to be unique. It is easy to see that the Polynomial Reconstruction problem and the Reed-Solomon decoding problem are equivalent. Both problems can be solved in polynomial time when $w \leq (n-k)/2$, using the Berlekamp-Welsh algorithm [3].

2.2 Augot and Finiasz' Cryptosystem

In the following, we briefly review Augot and Finiasz public-key cryptosystem [1].

Parameters: q is the size of F_q, n is the length of the Reed-Solomon code, k its dimension, W is the weight of a large error, so that the PR problem for n, k, W is believed to be hard, *i.e.* we must have:

$$W > \frac{n-k}{2}$$

ω is the weight of a small error, for which the PR problem with $n - W$ coordinates is easy:

$$\omega \leq \frac{n-W-k}{2} \tag{1}$$

It is recommended in [1] to take $n = 1024$, $k = 900$, $\omega = 25$, $W = 74$ and $q = 2^{80}$.

Key generation: Generate a unitary polynomial p of degree $k-1$, and a random n-dimensional vector E of weight W. Compute the codeword $c = ev(p)$ of RS_k. The public key is $z = c + E$, while the private key is (p, E).

Encryption: Let m a message of length $k-1$ over the alphabet F_q. The message m is seen as a polynomial $m(X) = m_0 + m_1 X + \ldots + m_{k-1}X^{k-2}$ of degree at most $k-2$. Generate a random $\alpha \in F_q$ and a random error e of weight ω. The ciphertext y is then:

$$y = ev(m) + \alpha \times (c + E) + e$$

Decryption: One considers only the positions where $E_i = 0$ and define the shortened code of length $n - W$, which is also a Reed-Solomon code of dimension k, which we denote $\overline{RS}_k$. Let $\overline{y}, \overline{ev}(m), \overline{c}, \overline{e}$ be the shortened $y, ev(m), c, e$. One must solve the equation:

$$\overline{y} = \overline{ev}(m) + \alpha \times \overline{c} + \overline{e}$$

We have $\overline{ev}(m) + \alpha \times \overline{c} \in \overline{RS}_k$, and from (1), the weight of the small error $\overline{e}$ is less than the error correction capacity of $\overline{RS}_k$; therefore, using the Berlekamp-Welsh algorithm, one can recover the unique polynomial r of degree $k-1$ such that:

$$ev(r) = \overline{ev}(m) + \alpha \times \overline{c}$$

which gives

$$r = m + \alpha \cdot p$$

Since $\deg(m) \leq k-2$ and p is a unitary polynomial of degree $k-1$, the field element α is the leading coefficient of r. Therefore one can recover m as:

$$m = r - \alpha \cdot p$$

3 Our Attack

The attack is a variant of the Berlekamp-Welsh algorithm for solving the PR problem (see [6]).

Let n, k, W, ω and q be the parameters of the system. Let (p, E) be the private key and $z = ev(p) + E$ be the public-key. Let m be the plaintext encoded as a polynomial of degree less than $k-2$. Let e be an error vector of weight ω, and α be a field element. Let

$$y = ev(m) + \alpha \times z + e \tag{2}$$

be the corresponding ciphertext.

Theorem 1. *Given the public-key z and the ciphertext y, one can recover the corresponding plaintext m in polynomial time.*

Proof. Let y_i, z_i and e_i be the components of the words y, z and e. Given y and z, one must solve the following set of equations:

$$\exists e, m, \alpha,\ y_i = m(x_i) + \alpha \cdot z_i + e_i \ \text{ for all } 1 \leq i \leq n \tag{3}$$

where the weight of e is less than ω. Note that from the definition of the cryptosystem, there is a unique solution.

Consider the following set of equations:

$$\exists V, m, \alpha,\ \begin{cases} \deg(V) \leq \omega,\ V \neq 0,\ \deg(m) \leq k-2 \\ \forall i,\ \ V(x_i) \cdot (y_i - \alpha \cdot z_i) = V(x_i) \cdot m(x_i) \end{cases} \tag{4}$$

Any solution V, m, α of (4) gives a solution to (3). Namely, the fact that $V \neq 0$ and $\deg V \leq \omega$ implies that V can be equal to zero at most ω times. Therefore, letting $e_i = y_i - m(x_i) - \alpha \cdot z_i$, the weight of e is less than ω.

Conversely, any solution to (3) gives a solution to (4). Namely, one can take $V(X) = \prod_{i \in B}(X - x_i)$ with $B = \{i | e_i \neq 0\}$. The problem of solving (3) can thus be reduced to finding V, m, α satisfying (4). Consider now the following set of equations:

$$\exists V, N, \lambda,\ \begin{cases} \deg(V) \leq \omega,\ V \neq 0,\ \deg(N) \leq k + \omega - 1 \\ \forall i,\ \ V(x_i) \cdot (y_i - \lambda \cdot z_i) = N(x_i) \end{cases} \tag{5}$$

The system (5) is a linearized version of (4), in which one has replaced the product $V(x_i) \cdot m(x_i)$ by $N(x_i)$. It is easy to see that any solution of (4) gives a solution to (5), as one can take $\lambda = \alpha$ and $N = m \cdot V$. However, the converse is not necessarily true.

For a given λ, the system (5) gives a linear system of n equations in the $k + 2 \cdot \omega + 1$ unknown, which are the coefficients of the polynomials V and N. More precisely, denoting:

$$V(X) = \sum_{i=0}^{\omega} v_i \cdot X^i, \quad N(X) = \sum_{i=0}^{k+\omega-1} n_i \cdot X^i$$

and Y the vector of coordinates:

$$Y = (v_0, \cdots, v_\omega, n_0, \cdots, n_{k+\omega-1})$$

one let $M(\lambda)$ be the matrix of the system:

$$M(\lambda)_{i,j} = \begin{cases} (y_i - \lambda \cdot z_i) \cdot (x_i)^j & \text{if } 0 \leq j \leq \omega \\ -(x_i)^{j-\omega-1} & \text{if } \omega < j < k + 2\omega + 1 \end{cases}$$

The matrix $M(\lambda)$ is a rectangular matrix with n lines and $k + 2\omega + 1$ columns; from (1) we have that $n > k + 2\omega + 1$. The coefficients of $M(\lambda)$ are a function of the public-key and the ciphertext only. The system (5) is then equivalent to:

$$\exists Y, \lambda, \quad M(\lambda).Y = 0, \ Y \neq 0 \tag{6}$$

We consider the matrix $M(\lambda)$ with $\lambda = 0$. Using Gaussian elimination, we compute the rank of the matrix $M(0)$. We distinguishe two cases: $rank\ M(0) = k + 2\omega + 1$, and $rank\ M(0) < k + 2\omega + 1$.

If $rank\ M(0) = k + 2\omega + 1$, then there exists a square sub-matrix of $M(0)$ of dimension $k + 2\omega + 1$ which is invertible. Without loss of generality, one can assume that the matrix obtained by taking the first $k + 2\omega + 1$ lines of $M(0)$ is invertible. Let $M'(\lambda)$ be the square matrix obtained by taking the first $k + 2\omega + 1$ lines of $M(\lambda)$. Any solution Y, λ of (6) satisfies:

$$M'(\lambda).Y = 0, \ Y \neq 0$$

which implies that the matrix $M'(\lambda)$ is non-invertible, *i.e.* $\det(M(\lambda)) = 0$. Then, the solution α in system (4) must be a root of the function:

$$f(\lambda) = \mathrm{Det}(M'(\lambda))$$

which is a polynomial of degree at most $\omega + 1$. The polynomial f is not identically zero, because $M'(0)$ is invertible, which implies $f(0) \neq 0$. The polynomial f can easily be obtained from the public-key z and the ciphertext y by computing $f(\lambda) = \mathrm{Det}(M'(\lambda))$ for $\omega + 2$ distinct values of λ and then using Lagrange interpolation.

The factorization of a polynomial over a finite-field can be done in polynomial time (see for example [13]). Therefore, one obtains a list of at most $\omega + 1$ candidates, one of which being the solution α of (4), and equivalently, of (3). For the right candidate α, the vector $y - \alpha \times z$ is equal to $ev(m) + e$, where the weight of e is less than the error correcting capacity of the Reed-Solomon code. Therefore, using Berlekamp-Welsh algorithm, one recovers the plaintext m from $y - \alpha \times z$ in polynomial time.

More precisely, let α, m, e be the solution of (3). Given a solution V, N, λ of (5) with $\lambda = \alpha$, we have for all $1 \leq i \leq k + 2 \cdot \omega + 1$:

$$V(x_i) \cdot (m(x_i) + e_i) = N(x_i)$$

Since the error vector e has a weight at most ω, we have for at least $\omega + k + 1$ values of i:

$$V(x_i) \cdot m(x_i) = N(x_i)$$

N and $V \cdot m$ are therefore two polynomials of degree less than $\omega + k - 1$ which take the same value on at least $\omega + k + 1$ distinct points; consequently, the two polynomials must be equal. This means that one can recover m by performing a polynomial division:

$$m = \frac{N}{V}$$

Therefore, one can recover the plaintext in polynomial time.

Let us now consider the second case, *i.e.* $rank\ M(0) < k + 2\omega + 1$. Then there exists $Y \neq 0$ such that $M(0).Y = 0$. The vector Y gives the coefficients of two polynomials V and N such that for all $1 \leq i \leq n$:

$$V(x_i) \cdot y_i = N(x_i)$$

From (2) we have $y_i = m(x_i) + \alpha \cdot (p(x_i) + E_i) + e_i$, which gives for all i:

$$V(x_i) \cdot \big((m + \alpha \cdot p)(x_i) + \alpha \cdot E_i + e_i\big) = N(x_i)$$

The weight of E is at most W and the weight of e is at most ω. Moreover, from (1) we have $n \geq k + 2\omega + W$. Therefore, for at least $\omega + k$ values of i, we have:

$$V(x_i) \cdot (m + \alpha \cdot p)(x_i) = N(x_i)$$

As previously, $V \cdot (m + \alpha \cdot p)$ and N are two polynomials of degree less than $k + \omega - 1$ which take the same value on at least $\omega + k$ distinct points; consequently, they must be equal, which gives:

$$m + \alpha \cdot p = \frac{N}{V}$$

Since the polynomial p is unitary and $\deg p = k - 1$ and $\deg m \leq k - 2$, this enables to recover α. Then, as previously, given α, we recover m in polynomial time[1]. □

4 The Repaired Cryptosystem

In this section, we describe the repaired cryptosystem published in [2]. The new cryptosystem is resistant against the previous attack. The reparation is based on working in the subfield of a given field, and using the trace operator. Following [2], we recall these notions in the next section.

4.1 Subfields and Trace Operator

We consider the finite field $\mathrm{GF}(q^u)$, where q is the power of a prime integer. The finite field $\mathrm{GF}(q)$ is a subfield of $\mathrm{GF}(q^u)$. The finite field $\mathrm{GF}(q^u)$ can be viewed as a u-dimensional vector space over $\mathrm{GF}(q)$. Let $\gamma_1, \ldots, \gamma_u$ be a basis of $\mathrm{GF}(q^u)$ over $\mathrm{GF}(q)$, then every element $\alpha \in \mathrm{GF}(q^u)$ can be uniquely written $\alpha = \sum_{i=1}^{u} \alpha_i \gamma_i$, where $\alpha_i \in \mathrm{GF}(q)$.

Definition 2. *The trace operator of $GF(q^u)$ into $GF(q)$ is defined by:*

$$\forall x \in GF(q^u),\ Tr(x) = x + x^q + \ldots + x^{q^{u-1}}$$

The trace operator is a $\mathrm{GF}(q)$-linear mapping (and not $\mathrm{GF}(q^u)$-linear) of $\mathrm{GF}(q^u)$ into $\mathrm{GF}(q)$. For any basis $\gamma_1, \ldots, \gamma_u$ of $\mathrm{GF}(q^u)$, there exists a unique dual basis $\gamma_1^*, \ldots, \gamma_u^*$ with respect to the Trace operator. The dual basis is such that:

$$\mathrm{Tr}(\gamma_i \gamma_j^*) = 1 \text{ if } i = j, \text{ and } 0 \text{ otherwise}$$

The dual basis can be efficiently computed.

We extend the trace operator to vectors:

$$\mathrm{Tr}(c_1, \ldots, c_n) = (\mathrm{Tr}(c_1), \ldots, \mathrm{Tr}(c_n))$$

[1] In this second case, we can also recover the private key (p, E). It has been shown in [9] that this second case happens with negligible probability.

and to polynomials: for any polynomial $p \in \mathrm{GF}(q^u)[X]$, $p(x) = \sum_{i=0}^{k} p_i x^i$, we define the polynomial $\mathrm{Tr}(p) \in \mathrm{GF}(q)[X]$ as:

$$\mathrm{Tr}(p)(x) = \sum_{i=0}^{k} \mathrm{Tr}(p_i) x^i$$

Let $x_1, \cdots, x_n$ be n distinct elements of $\mathrm{GF}(q) \in \mathrm{GF}(q^u)$. As in section 2.1 we denote by ev the following map:

$$ev : \begin{cases} \mathrm{GF}(q^u)[X] \to \mathrm{GF}(q^u)^n \\ p(X) \qquad\quad \to (p(x_1), \ldots, p(x_n)) \end{cases}$$

Proposition 1. *For all $p \in GF(q^u)[X]$, we have $Tr(ev(p)) = ev(Tr(p))$*

Proof. The j-th component of $\mathrm{Tr}(ev(p))$ is

$$\mathrm{Tr}(p(x_j)) = \mathrm{Tr}(\sum_{i=0}^{k} p_i \cdot (x_j)^i)$$

From the $\mathrm{GF}(q)$-linearity of the Trace operator and the fact that $x_j \in \mathrm{GF}(q)$, we obtain:

$$\mathrm{Tr}(p(x_j)) = \sum_{i=0}^{k} \mathrm{Tr}(p_i)(x_j)^i$$

which is the j-th component of $ev(\mathrm{Tr}(p))$. □

As in section 2.1, we define the Reed-Solomon code of dimension k and length n over $\mathrm{GF}(q^u)$ as the following set of n-tuples (codewords):

$$RS_k = \{ev(f); f \in \mathrm{GF}(q^u)[X], \deg f < k\}$$

4.2 The Repaired Cryptosystem

In this section, we recall the repaired cryptosystem [2].

Parameters: A finite field $\mathrm{GF}(q^u)$, an integer n as the length of the Reed-Solomon code, k its dimension, W is the weight of a large error, ω is the weight of a small error, for which the PR problem with $n - W$ coordinates is easy:

$$\omega \leq \frac{n - W - k}{2} \tag{7}$$

The authors of the repaired cryptosystem recommend in [2] to take $q = 2^{20}$, $u = 4$, $n = 2048$, $k = 1400$, $W = 546$ and $\omega = 49$. [2]

[2] Actually, the authors of [2] forgot to clearly specify k, but they state that with these parameters, "a plaintext consists of $k - u$ elements in $\mathrm{GF}(2^{20})$, that is 27920 bits", from which we infer that $k = 27920/20 + 4 = 1400$

Key generation: Generate a random polynomial p of degree $k-1$ over $\mathrm{GF}(q^u)$, such that the u coefficients $p_{k-1}, \ldots, p_{k-u}$ form a basis of $\mathrm{GF}(q^u)$ over $\mathrm{GF}(q)$. Compute $c = ev(p) \in RS_k$. Generate a random n-dimensional vector E of weight W with coefficients in $\mathrm{GF}(q^u)$. The public-key is the vector $K = c + E$ over $\mathrm{GF}(q^u)$. The private key is (p, E).

Encryption: Let m a message of length $k-u$ over the alphabet $\mathrm{GF}(q)$. The message m is seen as a polynomial $m(X) = m_0 + m_1 X + \ldots + m_{k-1} X^{k-u-1}$ in $\mathrm{GF}(q)[X]$. Generate a random $\alpha \in \mathrm{GF}(q^u)$ and a random vector e of weight ω over $\mathrm{GF}(q)$. The ciphertext y is then:

$$y = ev(m) + \mathrm{Tr}(\alpha \cdot K) + e$$

Decryption: One considers only the positions where $E_i = 0$ and define the shortened code of length $n - W$, which is also a Reed-Solomon code of dimension k, which we denote $\overline{RS}_k$. Let $\overline{y}, \overline{c}, \overline{e}$ be the shortened y, c, e and let $\overline{ev}$ be the shortened map ev. One must solve the equation:

$$\overline{y} = \overline{ev}(m) + \mathrm{Tr}(\alpha \cdot \overline{c}) + \overline{e}$$

Using proposition 1, we have:

$$\mathrm{Tr}(\alpha \cdot \overline{c}) = \mathrm{Tr}(\alpha \cdot \overline{ev}(p)) = \mathrm{Tr}(\overline{ev}(\alpha p)) = \overline{ev}(\mathrm{Tr}(\alpha p))$$

Thus $\overline{ev}(m) + \mathrm{Tr}(\alpha \cdot \overline{c}) = \overline{ev}(m + \mathrm{Tr}(\alpha p)) \in \overline{RS}_k$, and from (7), the weight of the small error $\overline{e}$ is less than the error correction capacity of $\overline{RS}_k$; therefore, using the Berlekamp-Welsh algorithm, one can recover the polynomial $q = m + \mathrm{Tr}(\alpha p)$.

Letting $q = \sum_{i=0}^{k-1} q_i x^i$, since $\deg(m) \leq k - u - 1$, we have $q_i = \mathrm{Tr}(\alpha p_i)$ for $i = k-u, \ldots, k-1$. This gives the u coordinates of α in the dual basis of $p_{k-u}, \ldots, p_{k-1}$, from which we derive α. From α one recovers m as $m = q - \mathrm{Tr}(\alpha p)$.

5 The Attack against the Repaired Cryptosystem

In this section, we describe an attack that breaks the repaired cryptosystem. Given the public key and a ciphertext, we recover the plaintext in polynomial time. As the attack of section 3, it is a variant of the Berlekamp-Welsh algorithm, but as opposed to the previous attack, it is only a heuristic (but it works very well in practice).

Let $\mathrm{GF}(q^u)$, n, k, W, ω be the parameters of the system. Let (p, E) be the private key and $K = ev(p) + E$ be the public-key. Let m be the plaintext encoded as a polynomial of degree less than $k - u - 1$. Let e be an error vector of weight ω, and $\alpha \in \mathrm{GF}(q^u)$. Let

$$y = ev(m) + \mathrm{Tr}(\alpha \cdot K) + e$$

be the corresponding ciphertext.

Let $\gamma_1, \ldots, \gamma_u$ be a basis of $\mathrm{GF}(q^u)$ over $\mathrm{GF}(q)$. We write $\alpha = \sum_{t=1}^{u} \alpha_t \cdot \gamma_t$ where $\alpha_t \in \mathrm{GF}(q)$. We have

$$\mathrm{Tr}(\alpha \cdot K) = \sum_{t=1}^{u} \alpha_t \mathrm{Tr}(\gamma_t \cdot K)$$

For $t = 1, \ldots, u$, we define:

$$K_t = \mathrm{Tr}(\gamma_t \cdot K)$$

Note that the u vectors K_t are vectors over $\mathrm{GF}(q)$ which can be computed from the public-key K. Finally the ciphertext can be written as:

$$y = ev(m) + \sum_{t=1}^{u} \alpha_t \cdot K_t + e \tag{8}$$

Note that in equation (8), all computation is done in the subfield $\mathrm{GF}(q)$. Let $y_i, K_{t,i}$ and e_i be the components of the vectors y, K_t and e. Given y and K_t, one must solve the following set of equations:

$$\exists e, m, \alpha_1, \ldots, \alpha_u, \ y_i = m(x_i) + \sum_{t=1}^{u} \alpha_t \cdot K_{t,i} + e_i \ \text{ for all } 1 \leq i \leq n \tag{9}$$

where the weight of e is ω. Note that from the definition of the cryptosystem, there is a unique solution.

Let V, $R_1, \ldots, R_u$ be polynomials of degree at most ω, with $V \neq 0$. Let N be a polynomial of degree at most $\omega + k - u - 1$. Consider the following set of equations, where the unknown are the polynomials V, $R_1, \ldots, R_u$ and N:

$$\forall i \in [1, n], \quad V(x_i) \cdot y_i = N(x_i) + \sum_{t=1}^{u} K_{t,i} \cdot R_t(x_i) \tag{10}$$

It is clear that given a solution to system (9), one can obtain a solution to system (10) with $V \neq 0$. Namely, one can take $V(X) = \prod_{i \in B}(X - x_i)$ with $B = \{i | e_i \neq 0\}$, and $R_t = \alpha_t \cdot V$ for $t = 1, \ldots, u$, and $N = m \cdot V$. This shows that the system (10) has at least a non-zero solution.

The system (10) gives a homogeneous linear system of n equations in the $k+(u+2)\cdot\omega+1$ unknowns, which are the coefficients of the polynomials V, $R_1, \ldots, R_u$ and N. Let M be the matrix of the corresponding system. The matrix has $k + (u + 2) \cdot \omega + 1$ columns and n rows and can be computed from the ciphertext and the public-key. In the following, we assume that:

$$n \geq k + (u + 2) \cdot \omega \tag{11}$$

This inequality is valid for the proposed parameters. Since the system (10) has at least a non-zero solution, the matrix cannot be of maximum rank, therefore $rank\ M \leq k + (u + 2) \cdot \omega$.

In the following, we assume that $rank\ M = k + (u + 2) \cdot \omega$. This is the only assumption that we make for our cryptanalysis. It seems that in practice, this assumption is always satisfied. In this case, the kernel of M is a linear space of dimension 1. We have already seen that $V(X) = \prod_{i \in B}(X - x_i)$ with $B = \{i | e_i \neq 0\}$, and $R_t = \alpha_t \cdot V$ for $t = 1, \ldots, u$ and $N = m \cdot V$ is a solution to the system (10), and so $(V, R_1, \ldots, R_t, N)$ generates the kernel of M.

Therefore, if we compute by Gaussian elimination an element $(V', R'_1, \ldots, R'_u, N')$ in $\ker M$, we must have that $V' = \lambda \cdot V$, $R'_t = \lambda R_t$ for $t = 1, \ldots, u$ and $N' = \lambda \cdot N$ for some

$\lambda \in \mathrm{GF}(q)$ with $\lambda \neq 0$. Therefore, we have $N' = \lambda \cdot N = \lambda \cdot m \cdot V = m \cdot V'$ and we can recover m by doing a polynomial division:

$$m = \frac{N'}{V'}$$

To summarize, assuming that $rank\ M = k + (u+2) \cdot \omega$, we recover the plaintext from the public-key and the ciphertext in polynomial time.

6 Practical Experiments

In appendix, we illustrate the attack against the original Augot and Finiasz' cryptosystem for small parameters. We have also implemented our attack using Shoup's NTL library [12]. The attack works well in practice. For the recommended parameters ($n = 1024$, $k = 900$, $\omega = 25$, $W = 74$, $q = 2^{80}$), it takes roughly 30 minutes on a single PC to recover the plaintext from the ciphertext and the public-key. We have also implemented our attack against the repaired cryptosystem, and for the recommended parameters, it takes roughly 8 minutes on a single PC to recover the plaintext from the ciphertext and the public-key.

7 Discussion

In this section, we try to see if it is possible to modify the parameters of the scheme in order to resist to the previous attack. The only condition on the parameters for the attack to work is inequality (11). Therefore, one may try to increase k, u or ω while keeping n constant. In the following, we show that this is not possible. Namely, we describe another attack on the repaired cryptosystem that recovers the private-key from the public-key. The attack does not work for the recommended parameters, but applies for large u.

The attack is the following. Let $K = ev(p) + E$ be the public-key with the n components K_i, where $\deg p = k - 1$ and the weight of E is W. The Berlekamp-Welsh algorithm for recovering p from K is the following: it looks for two polynomials V and N such that $\deg V = W$, $\deg N = k + W - 1$ and $V \neq 0$, such that:

$$\forall i \in [1, n], V(x_i) \cdot K_i = N(x_i)$$

This gives a homogeneous linear system of n equations in $k + 2 \cdot W + 1$ unknown. This system has a non-zero solution as we can take $V(X) = \prod_{i \in B}(X - x_i)$ with $B = \{i | E_i \neq 0\}$ and $N = p \cdot V$. Letting V, N be any non-zero solution, we have for at least $n - W$ values of i:

$$V(x_i) \cdot p(x_i) = N(x_i)$$

Therefore, if $n - W > k + W - 1$, or equivalently,

$$n \geq k + 2 \cdot W \tag{12}$$

the polynomials $V \cdot p$ and N must be equal, which enables to recover p as $p = N/V$.

As in the attack of section 5, from K we derive the u vectors K_t for $t = 1, \ldots, u$ such that:

$$K_t = \mathrm{Tr}(\gamma_t \cdot K)$$

where $\gamma_1, \ldots, \gamma_u$ is a basis of $\mathrm{GF}(q^u)$ over $\mathrm{GF}(q)$. Then we have:

$$K_t = \mathrm{Tr}(\gamma_t \cdot (ev(p) + E)) = ev(\mathrm{Tr}(\gamma_t \cdot p)) + \mathrm{Tr}(\gamma_t \cdot E)$$

Letting $p_t = \mathrm{Tr}(\gamma_t \cdot p)$ and $E_t = \mathrm{Tr}(\gamma_t \cdot E)$, we can write:

$$\forall t \in [1, u], K_t = ev(p_t) + E_t$$

Therefore, we obtain a set of u vectors K_t which are evaluation of a polynomial p_t plus some error E_t. Thus we obtain u instances of the polynomial reconstruction problem over $\mathrm{GF}(q)$.

The key observation is that the instances are not independent because the errors occur is the same positions in all vectors E_t. This enables to derive the following improved attack: we look for a polynomial $V \neq 0$, $\deg V \leq W$ and polynomials $N_1, \ldots, N_u$, $\deg N_t \leq k + W - 1$ such that:

$$\forall i \in [1, n], \begin{cases} V(x_i) \cdot K_{1,i} = N_1(x_i) \\ \ldots \\ V(x_i) \cdot K_{u,i} = N_u(x_i) \end{cases}$$

We can take the same polynomial V for each $t \in [1, u]$ because the errors are in the same positions for all E_t. This gives a system of $u \cdot n$ equations in the $u \cdot k + (u + 1) \cdot W + 1$ unknowns. Let M be the corresponding matrix. It has $u \cdot n$ rows and $u \cdot k + (u+1) \cdot W + 1$ columns. We assume that:

$$u \cdot n \geq u \cdot k + (u + 1) \cdot W \tag{13}$$

The system has a non-zero solution. Therefore, the matrix cannot be of maximum rank, therefore $rank\ M \leq u \cdot k + (u + 1) \cdot W$. In the following, we assume that $rank\ M = u \cdot k + (u+1) \cdot W$. This makes our attack heuristic, but the heuristic works well in practice. In this case, as in section 5, the kernel of M is a linear space of dimension one, and given a solution $(V, N_1, \ldots, N_u)$, one can recover the polynomials p_t as $p_t = N_t/V$ and then recover the private key (p, E). A similar approach was already used in [4] for the decoding of interleaved Reed-Solomon codes.

The inequality (13) gives the following condition for the attack to work:

$$n \geq k + \frac{u + 1}{u} \cdot W$$

which is an improvement over (12). Note that for the recommended parameters in [2], the attack does not apply. Therefore, to prevent this attack, one must have:

$$n < k + \frac{u + 1}{u} \cdot W \tag{14}$$

Then, combining inequality (14) with inequality (7) which is necessary to be able to decrypt, one must have:

$$n \geq k + 2 \cdot (u + 1) \cdot \omega$$

which shows that condition (11) of the attack of section 5 is always satisfied. Therefore, there is no set of parameters which makes the repaired cryptosystem secure against both attacks.

8 Conclusion

We have broken the cryptosystem published by Augot and Finiasz at Eurocrypt 2003 and its reparation in [2]. In both cases, our attack recover the plaintext from the ciphertext and the public-key in polynomial time. Moreover, both attack work well in practice, as for the recommended parameters, one recovers the plaintext in a few minutes on a single PC.

References

1. D. Augot and M. Finiasz, *A Public Key encryption scheme based on the Polynomial Reconstruction problem*, Proceedings of Eurocrypt 2003, LNCS vol. 2656, Springer-Verlag.
2. D. Augot, M. Finiasz and P. Loidreau, Using the Trace Operator to repair the Polynomial Reconstruction based Cryptosystem presented at Eurocrypt 2003, Cryptology ePrint Archive, Report 2003/209, 30 Sep 2003, `http://eprint.iacr.org/`.
3. E.R. Berlekamp and L.R. Welch, *Error correction for algebraic block codes*. US Patent 4 633 470, 1986.
4. D. Bleichenbacher, A. Kiayias and M. Yung, *Decoding of Interleaved Reed Solomon Codes over Noisy Data*, proceedings of ICALP 2003.
5. J.S. Coron, *Cryptanalysis of a public-key encryption scheme based on the polynomial reconstruction problem*, Cryptology ePrint Archive, Report 2003/036, 5 Mar 2003, `http://eprint.iacr.org/`.
6. P. Gemmell and M. Sudan, *Highly resilient correctors for multivariate polynomials*, Information Processing Letters, 43(4): 169–174, September 1992.
7. V. Guruswami and M. Sudan, *Improved decoding of Reed-Solomon and Algebraic-Geometric codes*, IEEE Transactions on Information Theory, 45 : 1757-1767, 1999.
8. A. Kiayias and M. Yung, *Cryptographic hardness based on the decoding of Reed-Solomon codes with applications*, Proceedings of ICALP 2002, LNCS 2380, pp 232-243, 2002.
9. A. Kiayias and M. Yung, *Cryptanalysis of the polynomial reconstruction based public-key cryptosystem of Eurocrypt 2003 in the optimal parameter setting*, available at `http://www.cse.uconn.edu/~akiayias/`.
10. M. Naor and B. Pinkas, *Oblivious transfer and polynomial evaluation*. In ACM, editor, STOC 99, pp 245-254, 1999.
11. I.S. Reed and G. Solomon, *Polynomial codes over certain finite fields*, J. SIAM, 8:300-304, 1960.
12. V. Shoup, *NTL: A Library for doing Number Theory (version 5.3.1)*, publicly available at `www.shoup.net`.
13. V. Shoup, *A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic*, in Proc. 1991 International Symposium on Symbolic and Algebraic Computation, pp. 14-21, 1991.

A A Toy Example

In this section we illustrate the attack for small parameters. We take $n = 8$, $k = 3$, $\omega = 1$, $W = 3$. We work modulo $q = 11$. We take $x_i = i$ for $i = 1, \ldots, 8$. We take:

$$p(x) = x^2 + 5x + 3$$
$$E = (0, 0, 4, 0, 7, 6, 0, 0)$$

for the private key. The public-key is:

$$z = ev(p) + E = (9, 6, 9, 6, 5, 9, 10, 8)$$

Let the message m be $m(x) = 8x + 2$. Let $\alpha = 7$ and $e = (0, 5, 0, 0, 0, 0, 0, 0)$. The ciphertext y is:

$$y = ev(m) + \alpha \times z + e = (7, 10, 1, 10, 0, 3, 7, 1)$$

The matrix $M(\lambda)$ is then:

$$M(\lambda) = \begin{bmatrix} 7-9\lambda & 7-9\lambda & 10 & 10 & 10 & 10 \\ 10-6\lambda & 9-\lambda & 10 & 9 & 7 & 3 \\ 1-9\lambda & 3-5\lambda & 10 & 8 & 2 & 6 \\ 10-6\lambda & 7-2\lambda & 10 & 7 & 6 & 2 \\ -5\lambda & -3\lambda & 10 & 6 & 8 & 7 \\ 3-9\lambda & 7-10\lambda & 10 & 5 & 8 & 4 \\ 7-10\lambda & 5-4\lambda & 10 & 4 & 6 & 9 \\ 1-8\lambda & 8-9\lambda & 10 & 3 & 2 & 5 \end{bmatrix}$$

The determinant $f(\lambda)$ of the matrix $M'(\lambda)$ obtained by taking the first 6 lines of $M(\lambda)$ is equal to:

$$f(\lambda) = \det M'(\lambda) = 3\lambda^2 + 5\lambda + 5$$

which factors modulo $q = 11$ into:

$$f(\lambda) = 3 \cdot (\lambda - 6) \cdot (\lambda - 7)$$

For $\lambda = 7$, the matrix $M'(7)$ is non-invertible. We solve the linear system and find that $Y = (8, 7, 5, 1, 1, 0)$ is such that $M(7).Y = 0$; this gives $V(x) = 7x+8$ and $N(x) = x^2+x+5$, which gives modulo $q = 11$:

$$m(x) = N(x)/V(x) = 8x + 2$$

Finding Small Roots of Bivariate Integer Polynomial Equations Revisited

Eurocrypt 2004

Jean-Sébastien Coron

Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
jean-sebastien.coron@gemplus.com

Abstract. At Eurocrypt '96, Coppersmith proposed an algorithm for finding small roots of bivariate integer polynomial equations, based on lattice reduction techniques. But the approach is difficult to understand. In this paper, we present a much simpler algorithm for solving the same problem. Our simplification is analogous to the simplification brought by Howgrave-Graham to Coppersmith's algorithm for finding small roots of univariate modular polynomial equations. As an application, we illustrate the new algorithm with the problem of finding the factors of $n = pq$ if we are given the high order $1/4 \log_2 n$ bits of p.

1 Introduction

An important application of lattice reduction found by Coppersmith in 1996 is finding small roots of low-degree polynomial equations [3, 4, 5]. This includes modular univariate polynomial equations, and bivariate integer equations.

The problem of solving univariate polynomial equations modulo an integer N of unknown factorization seems to be hard, as for some polynomials it is equivalent to the knowledge of the factorization of N. Moreover, the problem of inverting RSA, *i.e.* extracting e-th root modulo N, is a particular case of this problem. However, at Eurocrypt '96, Coppersmith showed that the problem of finding *small* roots is easy [3, 5], using the LLL lattice reduction algorithm [9]:

Theorem 1 (Coppersmith). *Given a monic polynomial $P(x)$ of degree δ, modulo an integer N of unknown factorization, one can find in time polynomial in $(\log N, 2^\delta)$ all integers x_0 such that $P(x_0) = 0 \bmod N$ and $|x_0| \leq N^{1/\delta}$.*

The algorithm can be extended to handle multivariate modular polynomial equations, but the extension is heuristic only. Coppersmith's algorithm has many applications in cryptography: cryptanalysis of RSA with small public exponent when some part of the message is known [5], cryptanalysis of RSA with private exponent d smaller than $N^{0.29}$ [1], polynomial-time factorization of $N = p^r q$ for large r [2], and even an improved security proof for OAEP with small public exponent [13] (see [12] for a nice survey).

Coppersmith's algorithm for solving univariate modular polynomial equations was further simplified by Howgrave-Graham in [7]. Apart from being simpler to understand and implement, a significant advantage of Howgrave-Graham's approach is the heuristic extension to multivariate modular polynomial: indeed, depending on the shape of the polynomial, there is much flexibility in selecting the parameters of the algorithm, and Howgrave-Graham's approach enables to easily derive the corresponding bound for the roots. This approach is actually used in all previously cited variants of Coppersmith's technique [1, 2].

Similarly, the problem of solving bivariate integer polynomial equations seems to be hard. Letting $p(x, y)$ be a polynomial in two variables with integer coefficients,

$$p(x, y) = \sum_{i,j} p_{i,j} \cdot x^i y^j$$

it consists in finding all integer pairs (x_0, y_0) such that $p(x_0, y_0) = 0$. We see that integer factorization is a special case as one can take $p(x, y) = N - x \cdot y$. However, at Eurocrypt '96, Coppersmith showed [4, 5] that using LLL, the problem of finding *small* roots of bivariate polynomial equations is easy:

Theorem 2. *Let $p(x, y)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let X and Y be upper bounds on the desired integer solution (x_0, y_0), and let $W = \max_{i,j} |p_{ij}| X^i Y^j$. If $XY < W^{2/(3\delta)}$, then in time polynomial in $(\log W, 2^\delta)$, one can find all integer pairs (x_0, y_0) such that $p(x_0, y_0) = 0$, $|x_0| \leq X$, and $|y_0| \leq Y$.*

Moreover, there can be improved bounds depending on the shape of the polynomial $p(x, y)$. For example, for a polynomial $p(x, y)$ of total degree δ in x and y, the bound is $XY < W^{1/\delta}$. As for the univariate modular case, the technique can be heuristically extended to more than two variables. An application of Coppersmith's algorithm for the bivariate integer polynomial case is to factor in polynomial-time an RSA-modulus $n = pq$ such that half of the least significant or most significants bits of p are known [5].

However, as noted in [6], the approach for the bivariate integer case is rather difficult to understand. This means that the algorithm is difficult to implement in practice, and that improved bounds depending on the shape of the polynomial are more difficult to derive. In particular, what makes the analysis harder is that one has to derive the determinant of lattices which are not full rank. The particular case of factoring $n = pq$ when half of the least significant or most significants bits of p are known, was further simplified by Howgrave-Graham in [7], but as noted in [6], this particular simplification does not seem to extend to the general case of bivariate polynomial equations. As suggested in [12], a simplification analogue to what has been obtained by Howgrave-Graham for the univariate modular case would be useful.

In this paper, we present a simple and efficient algorithm for finding small roots of bivariate integer polynomials. Our simplification is analogous to the simplification obtained by Howgrave-Graham for the univariate modular case. We apply lattice reduction to a full rank lattice that admits a natural triangular basis. It is then straightforward to derive the determinant and improved bounds depending on the shape of the polynomial; the heuristic extension to more than two variables is also simpler. However, our algorithm is slightly less efficient than Coppersmith's algorithm, because our algorithm has a polynomial-time complexity only if $XY < W^{2/3\delta - \varepsilon}$ for any fixed $\varepsilon > 0$, whereas Coppersmith's algorithm requires $XY < W^{2/3\delta}$, a slightly weaker condition. In section 7, we illustrate our algorithm with the problem of finding the factors of $n = pq$ if we are given the high order $1/4 \log_2 n$ bits of p, and show that our algorithm is rather efficient in practice.

2 Solving Bivariate Integer Equations: an Illustration

In this section, we first illustrate our technique with a bivariate integer polynomial of the form

$$p(x, y) = a + bx + cy + dxy,$$

with $a \neq 0$ and $d \neq 0$. We assume that $p(x, y)$ is irreducible and has a small root (x_0, y_0). Our goal is to recover (x_0, y_0). As in theorem 2, we let X, Y be some bound on x_0, y_0, that is we have $|x_0| \leq X$ and $|y_0| \leq Y$, and let $W = \max\{|a|, |b|X, |c|Y, |d|XY\}$. Moreover, given a polynomial $h(x, y) = \sum_{i,j} h_{ij} x^i y^j$, we define $\|h(x, y)\|^2 := \sum_{i,j} |h_{ij}|^2$ and $\|h(x, y)\|_\infty := \max_{i,j} |h_{ij}|$. Note that we have:

$$W = \|p(xX, yY)\|_\infty \tag{1}$$

First, we generate an integer n such that :

$$W \leq n < 2 \cdot W \tag{2}$$

and $\gcd(n, a) = 1$. One can take $n = W + ((1 - W) \bmod |a|)$. Then we define the polynomial:

$$\begin{aligned} q_{00}(x, y) &= a^{-1} p(x, y) \bmod n \\ &= 1 + b'x + c'y + d'xy \end{aligned}$$

We also consider the polynomials $q_{10}(x, y) = nx$, $q_{01}(x, y) = ny$ and $q_{11}(x, y) = nxy$. Note that for all four polynomials $q_{ij}(x, y)$, we have that $q_{ij}(x_0, y_0) = 0 \bmod n$.

We consider the four polynomials $\tilde{q}_{ij}(x, y) = q_{ij}(xX, yY)$; we are interested in finding a small linear integer combination of the polynomials $\tilde{q}_{ij}(x, y)$. Therefore, we consider the lattice generated by all linear integer combinations of the coefficient vectors of the $\tilde{q}_{ij}(x, y)$. A basis of the lattice is given by the following matrix L of row vectors:

$$L = \begin{bmatrix} 1 & b'X & c'Y & d'XY \\ & nX & & \\ & & nY & \\ & & & nXY \end{bmatrix}$$

We know that the LLL algorithm [9], given a lattice spanned by $(u_1, \ldots, u_\omega)$, finds in polynomial time a lattice vector b_1 such that $\|b_1\| \leq 2^{(\omega-1)/4} \det(L)^{1/\omega}$. More background on lattice reduction techniques will be given in the next section. With $\omega = 4$ and $\det L = n^3 (XY)^2$ we obtain in polynomial time a non-zero polynomial $h(x, y)$ such that

$$\|h(xX, yY)\| \leq 2 \cdot n^{3/4} (XY)^{1/2} \tag{3}$$

Note that we have $h(x_0, y_0) = 0 \bmod n$. The following lemma, due to Howgrave-Graham, shows that if the coefficients of $h(x, y)$ are sufficiently small, then the equality $h(x_0, y_0) = 0$ holds not only modulo n, but also over $\mathbb{Z}$.

Lemma 1 (Howgrave-Graham). *Let $h(x, y) \in \mathbb{Z}[x, y]$ which is a sum of at most ω monomials. Suppose that $h(x_0, y_0) = 0 \bmod n$ where $|x_0| \leq X$ and $|y_0| \leq Y$ and $\|h(xX, yY)\| < n/\sqrt{\omega}$. Then $h(x_0, y_0) = 0$ holds over the integers.*

Proof. We have:

$$\begin{aligned}|h(x_0,y_0)| &= \left|\sum h_{ij}x_0^i y_0^i\right| = \left|\sum h_{ij}X^iY^j\left(\frac{x_0}{X}\right)^i\left(\frac{y_0}{Y}\right)^j\right| \\ &\leq \sum\left|h_{ij}X^iY^j\left(\frac{x_0}{X}\right)^i\left(\frac{y_0}{Y}\right)^j\right| \leq \sum\left|h_{ij}X^iY^j\right| \\ &\leq \sqrt{\omega}\|h(xX,yY)\| < n\end{aligned}$$

Since $h(x_0,y_0) = 0 \bmod n$, this gives $h(x_0,y_0) = 0$. □

Assume now that:

$$XY < n^{1/2}/16 \tag{4}$$

Then inequality (3) gives:

$$\|h(xX,yY)\| < n/2 \tag{5}$$

which implies that $h(x_0,y_0) = 0$. Moreover, from (1), (2) and (5) we get:

$$\|h(xX,yY)\| < n/2 < W \leq \|p(xX,yY)\|_\infty \leq \|p(xX,yY)\|$$

This shows that $h(x,y)$ cannot be a multiple of $p(x,y)$. Namely, if $h(x,y)$ is a multiple of $p(x,y)$, then it follows from the definition of p and h that we must have $h(x,y) = \lambda \cdot p(x,y)$ with $\lambda \in \mathbb{Z}^*$. This would give $\|h(xX,yY)\| = |\lambda| \cdot \|p(xX,yY)\| \geq \|p(xX,yY)\|$, a contradiction.

Eventually, since $p(x,y)$ is irreducible and $h(x,y)$ is not a multiple of $p(x,y)$,

$$Q(x) = \text{Resultant}_y(h(x,y),p(x,y))$$

gives a non-zero integer polynomial such that $Q(x_0) = 0$. Using any standard root-finding algorithm, we can recover x_0, and finally y_0 by solving $p(x_0,y) = 0$. Using inequality (4) and $n \geq W$, this shows that if :

$$XY < \frac{W^{1/2}}{16}$$

one can find in time polynomial in $\log W$ all integer pairs (x_0,y_0) such that $p(x_0,y_0) = 0$, $|x_0| \leq X$, and $|y_0| \leq Y$.

This bound is weaker than the bound $XY < W^{2/3}$ given by theorem 2 for $\delta = 1$. We will see in section 4 that by adding more multiples of $p(x,y)$ into the lattice, we recover the desired bound.

3 Background on Lattices and Polynomials

3.1 The LLL Algorithm

Let $u_1,\ldots,u_\omega \in \mathbb{Z}^n$ be linearly independent vectors with $\omega \leq n$. The lattice L spanned by $< u_1,\ldots,u_\omega >$ consists of all integral linear combinations of $u_1,\ldots,u_\omega$, that is:

$$L = \Big\{ \sum_{i=1}^{\omega} n_i \cdot u_i |\ n_i \in \mathbb{Z} \Big\}$$

Such a set of vectors u_i's is called a lattice *basis*. All the bases have the same number of elements, called the *dimension* or *rank* of the lattice. We say that the lattice is full rank if $\omega = n$. Any two bases of the same lattice L are related by some integral matrix of determinant ± 1. Therefore, all the bases have the same Gramian determinant $\det_{1 \leq i,j \leq d} < u_i, u_j >$. One defines the *determinant* of the lattice as the square root of the Gramian determinant. If the lattice is full rank, then the determinant of L is equal to the absolute value of the determinant of the $\omega \times \omega$ matrix whose rows are the basis vectors $u_1, \ldots, u_\omega$.

The LLL algorithm [9] computes a short vector in a lattice :

Theorem 3 (LLL). *Let L be a lattice spanned by $(u_1, \ldots, u_\omega)$. The LLL algorithm, given $(u_1, \ldots, u_\omega)$, finds in polynomial time a vector b_1 such that:*

$$\|b_1\| \leq 2^{(\omega-1)/4} \det(L)^{1/\omega}$$

3.2 Bound on the Factors of Polynomials

We use the following notation: given a polynomial $h(x) = \sum_i h_i x^i$, we define $\|h\|^2 := \sum_i |h_i|^2$ and $\|h\|_\infty := \max_i |h_i|$. We use the same notations for bivariate polynomials, as defined in section 2. The following two lemmata will be useful in the next section:

Lemma 2. *Let $a(x, y)$ and $b(x, y)$ be two non-zero polynomials over $\mathbb{Z}$ of maximum degree d separately in x and y, such that $b(x, y)$ is a multiple of $a(x, y)$ in $\mathbb{Z}[x, y]$. Then:*

$$\|b\| \geq 2^{-(d+1)^2} \cdot \|a\|_\infty$$

Proof. The proof is based on the following result of Mignotte [11]: let $f(x)$ and $g(x)$ be two non-zero polynomials over the integers, such that $\deg f \leq k$ and f divides g in $\mathbb{Z}[X]$; then :

$$\|g\| \geq 2^{-k} \cdot \|f\|_\infty$$

Let $f(x) = a(x, x^{d+1})$. Then we have $\deg f \leq (d+1)^2$ and the polynomials $a(x, y)$ and $f(x)$ have the same list of non-zero coefficients, which gives $\|f\|_\infty = \|a\|_\infty$. Similarly, letting $g(x) = b(x, x^{d+1})$, we have $\|g\| = \|b\|$. Moreover $f(x)$ divides $g(x)$ in $\mathbb{Z}[x]$. Using the previous result of Mignotte, this proves lemma 2. □

Lemma 3. *Let $a(x, y)$ and $b(x, y)$ be as in lemma 2. Assume that $a(0, 0) \neq 0$ and $b(x, y)$ is divisible by a non-zero integer r such that $\gcd(r, a(0, 0)) = 1$. Then $b(x, y)$ is divisible by $r \cdot a(x, y)$ and:*

$$\|b\| \geq 2^{-(d+1)^2} \cdot |r| \cdot \|a\|_\infty$$

Proof. Let $\lambda(x, y)$ be the polynomial such that $a(x, y) \cdot \lambda(x, y) = b(x, y)$. We show that r divides $\lambda(x, y)$. Assume that this is not the case, and let λ_{ij} be a coefficient of $x^i y^j$ in $\lambda(x, y)$ not divisible by r. Take the smallest (i, j) for the lexicographic ordering. Then we have that $b_{ij} = \lambda_{ij} \cdot a(0, 0) \bmod r$, where b_{ij} is the coefficient of $x^i y^j$ in $b(x, y)$. Since $a(0, 0)$ is invertible modulo r and $b_{ij} = 0 \bmod r$, this gives a contradiction. This shows that $r \cdot a(x, y)$ divides $b(x, y)$. Applying the previous lemma to $r \cdot a(x, y)$ and $b(x, y)$, this terminates the proof. □

4 Finding Small Roots of Bivariate Integer Polynomials

We prove the following theorem:

Theorem 4. *Let $p(x,y)$ be an irreducible polynomial in two variables over $\mathbb{Z}$, of maximum degree δ in each variable separately. Let X and Y be upper bounds on the desired integer solution (x_0,y_0), and let $W = \max_{i,j} |p_{ij}|X^iY^j$. If for some $\varepsilon > 0$,*

$$XY < W^{2/(3\delta)-\varepsilon} \tag{6}$$

then in time polynomial in $(\log W, 2^\delta)$, one can find all integer pairs (x_0,y_0) such that $p(x_0,y_0) = 0$, $|x_0| \leq X$, and $|y_0| \leq Y$.

Proof. We write:

$$p(x,y) = \sum_{0 \leq i,j \leq \delta} p_{ij}x^iy^j$$

and let (x_0,y_0) be an integer root of $p(x,y)$. As previously we let

$$W = \|p(xX,yY)\|_\infty$$

First we assume that $p_{00} \neq 0$ and $\gcd(p_{00}, XY) = 1$. We will see in appendix A how to handle the general case.

We select an integer $k \geq 0$ and let $\omega = (\delta+k+1)^2$. We generate an integer u such that $\sqrt{\omega} \cdot 2^{-\omega} \cdot W \leq u < 2W$ and $\gcd(p_{00}, u) = 1$. As in section 2, one can take

$$u = W + ((1-W) \bmod |p_{00}|)\,.$$

We let $n = u \cdot (XY)^k$. We have that $\gcd(p_{00}, n) = 1$ and:

$$\sqrt{\omega} \cdot 2^{-\omega} \cdot (XY)^k \cdot W \leq n < 2 \cdot (XY)^k \cdot W \tag{7}$$

As in section 2, we must find a polynomial $h(x,y)$ such that $h(x_0,y_0) = 0$ and $h(x,y)$ is not a multiple of $p(x,y)$. We let $q(x,y)$ be the polynomial:

$$\begin{aligned} q(x,y) &= p_{00}^{-1} \cdot p(x,y) \bmod n \\ &= 1 + \sum_{(i,j)\neq(0,0)} a_{ij}x^iy^j \end{aligned}$$

For all $0 \leq i,j \leq k$, we form the polynomials:

$$q_{ij}(x,y) = x^iy^jX^{k-i}Y^{k-j}q(x,y)$$

For all $(i,j) \in [0,\delta+k]^2 \setminus [0,k]^2$, we also form the polynomials:

$$q_{ij}(x,y) = x^iy^jn$$

We consider the corresponding polynomials $\tilde{q}_{ij}(x,y) = q_{ij}(xX,yY)$. Note that for all $(i,j) \in [0,\delta+k]^2$, we have that $q_{ij}(x_0,y_0) = 0 \bmod n$, and the polynomial $\tilde{q}_{ij}(x,y)$ is divisible by $(XY)^k$.

Let $h(x,y)$ be a linear integer combination of the polynomials $q_{ij}(x,y)$; the polynomial $\tilde{h}(x,y) = h(xX,yX)$ is also a linear combination of the $\tilde{q}_{ij}(x,y)$ with the same integer coefficients. We have that $h(x_0,y_0) = 0 \bmod n$ and $(XY)^k$ divides $h(xX,yY)$. Moreover $h(x,y)$ has maximum degree $\delta + k$ independently in x and y, therefore it is the sum of at most ω monomials. As in section 2, we are interested in finding a polynomial $h(x,y)$ such that the coefficients of $h(xX,yY)$ are small enough, for the following two reasons:

1) if the coefficients of $h(xX,yY)$ are sufficiently small, then the equality $h(x_0,y_0) = 0$ holds not only modulo n, but also over $\mathbb{Z}$. From lemma 1, the condition is:

$$\|h(xX,yY)\| < \frac{n}{\sqrt{\omega}} \tag{8}$$

2) if the coefficients of $h(xX,yY)$ are sufficiently small, then $h(x,y)$ cannot be a multiple of $p(x,y)$. Using lemma 3, the condition is:

$$\|h(xX,yY)\| < 2^{-\omega} \cdot (XY)^k \cdot W \tag{9}$$

This condition is obtained by applying lemma 3 with $a(x,y) = p(xX,yY)$, $b(x,y) = h(xX,yY)$ and $r = (XY)^k$. Then we have $a(0,0) = p_{00} \neq 0$ and $\gcd(a(0,0),(XY)^k) = 1$. Under condition (9), $h(xX,yY)$ cannot be a multiple of $p(xX,yY)$ and therefore $h(x,y)$ cannot be a multiple of $p(x,y)$.

Using inequality (7), we obtain that the first condition (8) is satisfied whenever the second condition (9) is satisfied.

We form the lattice L spanned by the coefficients of the polynomials $\tilde{q}_{ij}(x,y)$. The polynomials $q_{ij}(x,y)$ have a maximum degree of $\delta + k$ separately in x and y; therefore, there are $(\delta+k+1)^2$ such coefficients. Moreover, there is a total of $(\delta+k+1)^2$ polynomials. This gives a full rank lattice of dimension $\omega = (\delta + k + 1)^2$. In figure 1, we illustrate the lattice for $\delta = 1$ and $k = 1$.

	1	x	y	xy	x^2	x^2y	y^2	xy^2	x^2y^2
XYq	XY	$a_{10}X^2Y$	$a_{01}XY^2$	$a_{11}X^2Y^2$					
Yxq		XY		$a_{01}XY^2$	$a_{10}X^2Y$	$a_{11}X^2Y^2$			
Xyq			XY	$a_{10}X^2Y$			$a_{01}XY^2$	$a_{11}X^2Y^2$	
xyq				XY		$a_{10}X^2Y$		$a_{01}XY^2$	$a_{11}X^2Y^2$
x^2n					X^2n				
x^2yn						X^2Yn			
y^2n							Y^2n		
xy^2n								XY^2n	
x^2y^2n									X^2Y^2n

Fig. 1. The lattice L for $\delta = 1$ and $k = 1$

It is easy to see that the coefficient vectors of the polynomials $\tilde{q}_{ij}(x,y)$ form a triangular basis of L. The determinant is then the product of the diagonal entries. For $0 \leq i,j \leq k$, the contribution of the polynomials $\tilde{q}_{ij}(x,y)$ to the determinant is given by:

$$\prod_{0 \leq i,j \leq k} (XY)^k = (XY)^{k(k+1)^2}$$

The contribution of the other polynomials $\tilde{q}_{ij}(x,y)$ is then:

$$\prod_{(i,j)\in[0,\delta+k]^2\setminus[0,k]^2} X^iY^jn = (XY)^{\frac{(\delta+k)(\delta+k+1)^2}{2}-\frac{k(k+1)^2}{2}} n^{\delta(\delta+2k+2)}$$

Therefore, the determinant of L is given by:

$$\det(L) = (XY)^{\frac{(\delta+k)(\delta+k+1)^2+k(k+1)^2}{2}} n^{\delta(\delta+2k+2)} \tag{10}$$

Using LLL (see theorem 3), we obtain in time polynomial in $(\log W, \omega)$ a non-zero polynomial $h(x,y)$ such that:

$$\|h(xX,yY)\| \leq 2^{(\omega-1)/4} \cdot \det(L)^{1/\omega} \tag{11}$$

Note that any vector in the lattice L has integer coefficients divisible by $(XY)^k$; this means that in practice, it is more efficient to apply LLL to the lattice $(XY)^{-k}L$.

From inequality (11) we obtain that the conditions (8) and (9) are satisfied when:

$$2^{(\omega-1)/4} \cdot \det(L)^{1/\omega} < 2^{-\omega} \cdot (XY)^k \cdot W \tag{12}$$

In this case, we have that $h(x_0,y_0) = 0$ and $h(x,y)$ is not a multiple of $p(x,y)$. Since $p(x,y)$ is irreducible,

$$Q(x) = \text{Resultant}_y(h(x,y), p(x,y))$$

gives a non-zero integer polynomial such that $Q(x_0) = 0$. Using any standard root-finding algorithm, we can recover x_0, and finally y_0 by solving $p(x_0,y) = 0$.

Using inequality (7), we obtain that inequality (12) is satisfied when:

$$XY < 2^{-\beta} W^{\alpha} \tag{13}$$

where

$$\alpha = \frac{2(k+1)^2}{(\delta+k)(\delta+k+1)^2 - k(k+1)^2} \tag{14}$$

$$\beta = \frac{10}{4} \cdot \frac{(\delta+k+1)^4 + (\delta+k+1)^2}{(\delta+k)(\delta+k+1)^2 - k(k+1)^2} \tag{15}$$

We have that for all $\delta \geq 1$ and $k \geq 0$:

$$\alpha \geq \frac{2}{3\delta} - \frac{2}{3 \cdot (k+1)} \tag{16}$$

and:

$$\beta \leq \frac{4k^2}{\delta} + 13 \cdot \delta \tag{17}$$

Then, taking $k = \lfloor 1/\varepsilon \rfloor$, we obtain from (13), (16) and (17) the following condition for XY:

$$XY < W^{2/(3\delta)-\varepsilon} \cdot 2^{-4/(\delta\cdot\varepsilon^2)-13\delta} \tag{18}$$

For an XY satisfying (18), we obtain a bivariate integer polynomial root-finding algorithm running in time polynomial in $(\log W, \delta, 1/\varepsilon)$.

For an XY satisfying the slightly weaker condition (6), we exhaustively search the high order $4/(\delta \cdot \varepsilon^2) + 13\delta$ bits of x_0, so that condition (18) applies, and for each possible value we use the algorithm described previously. For a fixed $\varepsilon > 0$, the running time is polynomial in $(\log W, 2^\delta)$. This terminates the proof of theorem 4. □

As in [5], the efficiency of our algorithm depends on the shape of the polynomial $p(x, y)$. The previous theorem applies when $p(x, y)$ has maximum degree δ separately in x and y. If we assume that $p(x, y)$ has a total degree δ in x and y, we obtain the following theorem, analogous to theorem 3 in [5] (the proof is given in appendix B).

Theorem 5. *Under the hypothesis of theorem 4, except that $p(x, y)$ has total degree δ, the appropriate bound is:*

$$XY < W^{1/\delta - \varepsilon}$$

5 Comparison with Coppersmith's Algorithm

We note that under the following condition, stronger than (6) :

$$XY < W^{2/(3\delta) - \varepsilon} \cdot 2^{-13\delta}$$

Coppersmith's algorithm is polynomial-time in $(\log W, \delta, 1/\varepsilon)$ (see [5], theorem 2), whereas our algorithm is polynomial-time in $(\log W, \delta)$ but exponential-time in $1/\varepsilon$. Coppersmith's algorithm is therefore more efficient than ours for small values of ε. This implies that under the following condition, weaker than (6) :

$$XY < W^{2/(3\delta)}$$

Coppersmith's algorithm is still polynomial in $(\log W, 2^\delta)$ (see [5], corollary 2), which is no longer the case for our algorithm.

6 Extension to More Variables

Our algorithm can be extended to solve integer polynomial equations with more than two variables. As for Coppersmith's algorithm, the extension is heuristic only.

Let $p(x, y, z)$ be a polynomial in three variables over the integers, of degree δ independently in x, y and z. Let (x_0, y_0, z_0) be an integer root of $p(x, y, z)$, with $|x_0| \leq X$, $|y_0| \leq Y$ and $|z_0| \leq Z$. Let ℓ be an integer ≥ 0. As for the bivariate case, we generate an integer n such that $n = 0 \bmod (XYZ)^\ell$, and a polynomial $q(x, y, z)$ such that $q(x_0, y_0, z_0) = 0 \bmod n$ and $q(0, 0, 0) = 1 \bmod n$. Then we consider the lattice L generated by all linear integer combinations of the polynomials $x^i y^j z^k X^{\ell-i} Y^{\ell-j} Z^{\ell-k} q(xX, yY, zZ)$ for $0 \leq i, j, k \leq \ell$ and the polynomials $(xX)^i (yY)^j (zZ)^k \cdot n$ for $(i, j, k) \in [0, \delta + \ell]^3 \setminus [0, \ell]^3$. If the ranges X, Y, Z are small enough, then by using LLL we are guaranteed to find a polynomial $h_1(x, y, z)$ such that $h_1(x_0, y_0, z_0) = 0$ over $\mathbb{Z}$ and $h_1(x, y, z)$ is not a multiple of $p(x, y, z)$. Unfortunately, this is not enough. For small enough ranges X, Y, Z, we can also obtain a second polynomial $h_2(x, y, z)$ satisfying the same property. This can be done by bounding the

norm of the second vector produced by LLL, as in [1, 8]. Then we could take the resultant between the three polynomials $p(x, y, z)$, $h_1(x, y, z)$ and $h_2(x, y, z)$ in order to obtain a polynomial $f(x)$ such that $f(x_0) = 0$. But we have no guarantee that the polynomials $h_1(x, y, z)$ and $h_2(x, y, z)$ will be algebraically independent, for example we might have $h_2(x, y, z) = x \cdot h_1(x, y, z)$. This makes the method heuristic only.

7 Practical experiments

An application of solving bivariate equations described in [5] is factoring an RSA modulus $n = pq$ when the high-order bits of p are known. Using our algorithm from theorem 4, we obtain the following theorem, whose proof is given in appendix C.

Theorem 6. *For any $\varepsilon > 0$, given $n = pq$ and the high-order $(1/4 + \varepsilon) \log_2 n$ bits of p, we can recover the factorization of n in time polynomial in $\log n$.*

By comparison, Coppersmith's algorithm provides a slightly better result since only the high-order $1/4 \log_2 n$ bits of p are required (see theorem 4, [5]). The result of practical experiments are summarized in table 2, using Shoup's NTL library [14]. It shows that our bivariate polynomial root-finding algorithm works well in practice.

N	bits of p given	lattice dimension	running time
512 bits	144 bits	25	35 sec
512 bits	141 bits	36	3 min
1024 bits	282 bits	36	20 min

Fig. 2. Running times for factoring $N = pq$ given the high-order bits of p, using our bivariate integer polynomial root finding algorithm on a 733 Mhz PC running under Linux.

We have also implemented the factorization of $n = pq$ with high-order bits known using the simplification of Howgrave-Graham [7]. Results are given in table 3. It shows that the simplification of Howgrave-Graham is much more efficient in practice. Namely, the factorization of a 1024-bit RSA modulus knowing the high-order 282 bits of p takes roughly 20 minutes using our bivariate polynomial root finding algorithm, and only one second using Howgrave-Graham's simplification. This is due to the fact that the Howgrave-Graham simplification enables to obtain a lattice with a lower dimension (but it applies only to the particular case of factoring with high-bits known, not to the general case of finding small roots of bivariate integer polynomials).

N	bits of p given	lattice dimension	running time
1024 bits	282 bits	11	1 sec
1024 bits	266 bits	25	1 min
1536 bits	396 bits	33	19 min

Fig. 3. Running times for factoring $N = pq$ given the high-order bits of p, using Howgrave-Graham's algorithm on a 733 Mhz PC running under Linux.

8 Conclusion

We have presented an algorithm for finding small roots of bivariate integer polynomials, simpler than Coppersmith's algorithm. The bivariate integer case is now as simple to analyze and implement as the univariate modular case. Our algorithm is asymptotically less efficient than Coppersmith's algorithm, but experiments show that it works well in practice; however, for the particular case of integer factorization with high-bits known, the Howgrave-Graham simplification appears to be more efficient.

References

1. D. Boneh and G. Durfee, *Crypanalysis of RSA with private key d less than* $N^{0.292}$, proceedings of Eurocrypt '99, vol. 1592, Lecture Notes in Computer Science.
2. D. Boneh, G. Durfee and N.A. Howgrave-Graham, *Factoring* $n = p^r q$ *for large* r, proceedings of Crypto '99, vol. 1666, Lecture Notes in Computer Science.
3. D. Coppersmith, *Finding a Small Root of a Univariate Modular Equation*, proceedings of Eurocrypt '96, vol. 1070, Lecture Notes in Computer Science.
4. D. Coppersmith, *Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known*, proceedings of Eurocrypt' 96, vol. 1070, Lecture Notes in Computer Science.
5. D. Coppersmith, *Small solutions to polynomial equations, and low exponent vulnerabilities.* J. of Cryptology, 10(4)233-260, 1997. Revised version of two articles of Eurocrypt '96.
6. D. Coppersmith, *Finding small solutions to small degree polynomials.* In Proc. of CALC '01, LNCS, Sptinger-Verlag, 2001.
7. N.A. Howgrave-Graham, *Finding small roots of univariate modular equations revisited.* In Cryptography and Coding, volume 1355 of LNCS. Springer Verlag, 1997.
8. C.S. Jutla, *On finding small solutions of modular multivariate polynomial equations.* Proceedings of Eurocrypt '98, Lecture Notes in Computer Science. vol. 1402.
9. A.K. Lenstra, H.W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients.* Mathematische Ann., 261:513-534, 1982.
10. U. Maurer, *Fast Generation of Prime Numbers and Secure Public-Key Cryptographic Parameters*, Journal of Cryptology, vol. 8, no. 3, pp. 123-155, 1995.
11. M. Mignotte, *An inequality about factors of polynomials.* Math Comp. 28, 1153-1157, 1974.
12. P.Q. Nguyen and J. Stern, *The two faces of lattices in cryptology.* Proceedings of CALC '01, LNCS vol. 2146.
13. V. Shoup, *OAEP reconsidered.* Proceedings of Crypto '01, vol. 2139, Lecture Notes in Computer Science.
14. V. Shoup, *Number Theory C++ Library (NTL) version 5.3.1.* Available online at `www.shoup.net`.

A Finding Small Roots in the General Case

The algorithm described in section 4 assumes that $p(0,0) \neq 0$ and that $\gcd(p(0,0), XY) = 1$. Here we show how to handle the general case.

If $p(0,0) = 0$, we use a simple change of variable to derive a polynomial $p^*(x,y)$ such that $p^*(0,0) \neq 0$. This is done as follows: we write p as $p(x,y) = x \cdot a_0(x) + y \cdot c(x,y)$, where a_0 is a polynomial of maximum degree $\delta - 1$. Since $p(x,y)$ is irreducible, we must have $a_0 \neq 0$. Since $\deg a_0 \leq \delta - 1$, there exists $0 < i \leq \delta$ such that $a_0(i) \neq 0$. Then $p(i,0) \neq 0$ and letting $p^*(x,y) = p(x+i,y)$, we obtain that $p^*(0,0) \neq 0$ and use $p^*(x,y)$ instead of $p(x,y)$.

If $\gcd(p(0,0), XY) \neq 1$, we generate two random primes X' and Y' such that $X < X' < 2X$ and $Y < Y' < 2Y$, and X' and Y' do not divide $p(0,0)$. This can be done in polynomial-time using the recursive prime generation algorithm described in [10]. We then use X', Y' instead of X, Y.

B Proof of Theorem 5

We use the same n and the same $q(x,y)$ as in section 4. We use the same polynomials $q_{ij}(x,y) = x^i y^j X^{k-i} Y^{k-j} q(x,y)$, but only for $0 \leq i+j \leq k$ (instead of $0 \leq i,j \leq k$). We also use the polynomials $q_{ij}(x,y) = x^i y^j n$ for $k < i+j \leq k+\delta$.

We obtain a full-rank lattice L of dimension $\omega = (k+\delta+1)(k+\delta+2)/2$, where the coefficient vectors of the polynomials $\tilde{q}_{ij}(x,y)$ form a triangular basis. The contribution of the polynomials $\tilde{q}_{ij}(x,y)$ for $0 \leq i+j \leq k$ to the determinant is given by:

$$\prod_{0 \leq i+j \leq k} (XY)^k = (XY)^{\frac{k(k+1)(k+2)}{2}}$$

and the contribution of the remaining polynomial is:

$$\prod_{k < i+j \leq k+\delta} X^i Y^j n = (XY)^{d \cdot (2+d^2+6k+3k^2+3d(1+k))/6} \cdot n^{d(3+d+2k)/2}$$

which gives:

$$\det L = (XY)^{\frac{3k(1+k)(2+k)+d(2+d^2+6k+3k^2+3d(1+k))}{6}} \cdot n^{d(3+d+2k)/2}$$

As before, the condition is:

$$2^{(\omega-1)/4} \det(L)^{1/\omega} < 2^{-(k+\delta+1)^2} \cdot (XY)^k \cdot W$$

from which we derive the following condition on XY:

$$XY < W^{(1/\delta)-\varepsilon} \cdot 2^{-4/(\delta\varepsilon^2)-13\delta}$$

for $\varepsilon = \mathcal{O}(1/k)$. As previously, we exhaustive search on the high-order $4/(\delta\varepsilon^2) + 13\delta$ bits of x_0, to obtain the bound:

$$XY < W^{(1/\delta)-\varepsilon}$$

while remaining polynomial-time in $(\log W, 2^\delta)$.

C Factoring with High-bits known: Proof of Theorem 6

Let $N = pq$ be an RSA-modulus and assume that we know that high-order $(1/4+\varepsilon)\log_2 N$ bits of p, for $\varepsilon > 0$. By division we also know the high-order $(1/4+\varepsilon)\log_2 N$ bits of q. We write:

$$p = p_0 + x_0 \quad q = q_0 + y_0$$
$$|x_0| < p_0 N^{-1/4-\varepsilon} = X \quad |y_0| < q_0 N^{-1/4-\varepsilon} = Y$$

where p_0 and q_0 are known and x_0 and y_0 are unknown. We define the polynomial:

$$p(x,y) = (p_0 + x) \cdot (q_0 + y) - N = (p_0 q_0 - N) + q_0 x + p_0 y + xy$$

We have that $p(x_0, y_0) = 0$ and:

$$W = \max(|p_0 q_0 - n|, q_0 X, p_0 Y, XY) > q_0 X > \frac{1}{2} N^{3/4-\varepsilon}$$

We have:

$$XY = p_0 q_0 N^{-1/2-2\varepsilon} < N^{1/2-2\varepsilon}$$

which gives:

$$XY < 2W^{2/3-\varepsilon}$$

so that by guessing one additional bit of x_0 we are under the conditions of theorem 4.

Deterministic Polynomial Time Equivalence of Computing the RSA Secret Key and Factoring

To appear in Journal of Cryptology

Jean-Sébastien Coron and Alexander May

University of Luxembourg
162a, avenue de la Faiencerie, L-1511 Luxembourg
coron@clipper.ens.fr

Faculty of Computer Science, Electrical Engineering and Mathematics
University of Paderborn
33102 Paderborn, Germany
alexx@uni-paderborn.de

Abstract. We address one of the most fundamental problems concerning the RSA cryptosystem: does the knowledge of the RSA public and secret key-pair (e, d) yield the factorization of $N = pq$ in polynomial time? It is well-known that there is a *probabilistic* polynomial time algorithm that on input (N, e, d) outputs the factors p and q. We present the first *deterministic* polynomial time algorithm that factors N given (e, d) provided that $e, d < \phi(N)$. Our approach is an application of Coppersmith's technique for finding small roots of univariate modular polynomials.

Keywords: RSA, Coppersmith's theorem.

1 Introduction

The most basic security requirement for a public key cryptosystem is that it should be hard to recover the secret key from the public key. To establish this property, one usually identifies a well-known hard problem P and shows that recovering the secret key from the public key is polynomial-time equivalent to solving P.

In this paper we consider the RSA cryptosystem [11]. We denote by $N = pq$ the modulus, product of two primes p and q of the same bit-size. Furthermore, we denote by e, d the public and private exponents, such that $e \cdot d = 1 \bmod \phi(N)$, where $\phi(N) = (p-1) \cdot (q-1)$ is Euler's totient function. The public key is then (N, e) and the secret key is (N, d).

It is well known that there exists a *probabilistic* polynomial time equivalence between computing d and factoring N. The proof is given in the original RSA paper by Rivest, Shamir and Adleman [11] and is based on a work by Miller [8].

In this paper, we show that the equivalence can actually be made deterministic, namely we present the first *deterministic* polynomial-time algorithm that on input (N, e, d) outputs the factors p and q, provided that $e \cdot d \leq N^2$. Since for standard RSA, the exponents e and d are defined modulo $\phi(N)$, we have that $ed < \phi(N)^2 < N^2$ as required. Our result is mainly of theoretical interest, since our deterministic algorithm is much less efficient than the probabilistic one. However, we also present an algorithm that recovers the factors p and q deterministically in time $\mathcal{O}(\log^2 N)$ when $e \cdot d \leq N^{\frac{3}{2}}$; this happens when e is small and $d < \phi(N)$, which is common in practice.

Our technique is a variant of Coppersmith's theorem for finding small roots of univariate polynomial equations [2]. Coppersmith's theorem is based on the LLL lattice reduction

algorithm [6], and has found numerous applications in cryptanalysis (see [10] for a survey). We use a variant in which one considers polynomials modulo an unknown integer (instead of the known modulus). This variant was introduced by Boneh *et al.* in [1] for factoring moduli of the form $p^r q$ in polynomial time for large r. This approach was also used by Howgrave-Graham in [5] to compute approximate integer common divisors. Our technique is actually a direct application of Howgrave-Graham's algorithm, but for completeness we also provide a full description of our algorithm.

This article is an extended version of a paper published by A. May [7] at Crypto 2004. The difference with [7] is that our analysis is based on univariate modular polynomials instead of bivariate integer polynomials, which leads to a simpler algorithm. Moreover, we generalize our analysis to the case of unbalanced prime factors p and q. Quite expectedly, we obtain that the upper bound on ed gets larger when the prime factors are more imbalanced. For example, if $p < N^{1/4}$, then the modulus N can be factored in polynomial time given (e, d) for $e \cdot d \leq N^{8/3}$ (instead of N^2 for prime factors of equal size).

2 Background on Lattices

Let $u_1, \ldots, u_\omega \in \mathbb{Z}^n$ be linearly independent vectors with $\omega \leq n$. The lattice L spanned by $<u_1, \ldots, u_\omega>$ consists of all integral linear combinations of $u_1, \ldots, u_\omega$, that is:

$$L = \left\{ \sum_{i=1}^{\omega} n_i \cdot u_i |\ n_i \in \mathbb{Z} \right\}$$

Such a set $\{u_1, \ldots, u_\omega\}$ of vectors is called a lattice *basis*. All the bases have the same number of elements, called the *dimension* or *rank* of the lattice. We say that the lattice is full rank if $\omega = n$. Any two bases of the same lattice can be transformed into each other by a multiplication with some integral matrix of determinant ± 1. Therefore, all the bases have the same Gramian determinant $\det_{1 \leq i,j \leq d} < u_i, u_j >$. One defines the *determinant* of the lattice as the square root of the Gramian determinant. If the lattice is full rank, then the determinant of L is equal to the absolute value of the determinant of the $\omega \times \omega$ matrix whose rows are the basis vectors $u_1, \ldots, u_\omega$.

The LLL algorithm [6] computes a short vector in a lattice :

Theorem 1 (LLL). *Let L be a lattice spanned by $(u_1, \ldots, u_\omega) \in \mathbb{Z}^n$, where the Euclidean norm of each of the vectors $u_1, \ldots, u_\omega$ is bounded by B. Given $(u_1, \ldots, u_\omega)$, the LLL algorithm finds a vector b_1 such that:*

$$\|b_1\| \leq 2^{(\omega-1)/4} \det(L)^{1/\omega}$$

in time $\mathcal{O}(\omega^5 n \log^3 B)$

In order to improve the complexity of our algorithm, we will use an improved version of LLL, called the L^2 algorithm and due to Nguyen and Stehlé [9]. The L^2 algorithm achieves the same bound on $\|b_1\|$ but in time $\mathcal{O}(\omega^4 n(\omega + \log B) \log B)$.

3 An Algorithm for $ed \leq N^{\frac{3}{2}}$

In this section, we consider the standard RSA setting, *i.e.* we assume that N is the product of two different prime factors p, q of the same bit-size. We also assume that $ed \leq N^{\frac{3}{2}}$. This is a practical case since for RSA one generally uses a small public exponent e (for example, $e = 3$ or $e = 2^{16} + 1$). The following theorem shows that the factorisation of N can then be recovered in deterministic time $\mathcal{O}(\log^2 N)$:

Theorem 2. *Let $N = p \cdot q$, where p and q are two prime integers of same bit-size. Let e, d be such that $e \cdot d = 1 \bmod \phi(N)$. Then if $1 < e \cdot d \leq N^{3/2}$, there is a deterministic algorithm that given (N, e, d) recovers the factorization of N in time $\mathcal{O}(\log^2 N)$.*

Proof. In the following, we assume wlog that $p < q$, which implies :

$$p < N^{\frac{1}{2}} < q < 2p < 2N^{\frac{1}{2}}.$$

This gives the following useful estimates:

$$p + q < 3N^{\frac{1}{2}} \quad \text{and} \quad \phi(N) = N + 1 - (p + q) > \frac{1}{2}N. \tag{1}$$

Let us denote by $\lceil k \rceil$ the smallest integer greater or equal to k. Furthermore, we denote by $\mathbb{Z}^*_{\phi(N)}$ the group of invertible integers modulo $\phi(N)$.

Since $ed = 1 \bmod \phi(N)$, we know that

$$ed = 1 + k\phi(N) \quad \text{for some } k \in \mathbb{N}.$$

We show that k can be recovered up to a small constant when $ed \leq N^{\frac{3}{2}}$. Namely, we define $\tilde{k} = \frac{ed-1}{N}$ as an underestimate of k and we observe that :

$$\begin{aligned} k - \tilde{k} &= \frac{ed-1}{\phi(N)} - \frac{ed-1}{N} \\ &= \frac{N(ed-1) - (N - p - q + 1)(ed-1)}{\phi(N)N} \\ &= \frac{(p+q-1)(ed-1)}{\phi(N)N} \end{aligned}$$

Using (1) we conclude that :

$$k - \tilde{k} < 6N^{-\frac{3}{2}}(ed - 1). \tag{2}$$

Then since $ed \leq N^{\frac{3}{2}}$, we obtain that $0 < k - \tilde{k} < 6$. Thus, one of the six values $\left\lceil \tilde{k} \right\rceil + i$, $i = 0, 1, \ldots, 5$ must be equal to k. We can test these six candidates successively and for the right choice k, we can compute :

$$N + 1 + \frac{1 - ed}{k} = p + q$$

from which one recovers the factorization of N. Our approach uses only elementary arithmetic on integers of bit-size $\mathcal{O}(\log(N))$. Thus, the running time is $\mathcal{O}(\log^2 N)$, which concludes the proof of the theorem. □

4 The Case of $ed \leq N^2$

As in the previous section, we assume that N is the product of two primes p and q of same bit-size, but here we only assume that $ed \leq N^2$. Under this assumption, we show the *deterministic* polynomial-time equivalence between recovering d and factoring N. We will generalize to an $N = pq$ with unbalanced prime factors in the next section.

Theorem 3. *Let $N = p \cdot q$, where p and q are two prime integers of same bit-size. Let e, d be such that $e \cdot d = 1 \bmod \phi(N)$. Then if $1 < e \cdot d \leq N^2$, there is a deterministic algorithm that given (N, e, d) recovers the factorization of N in time $\mathcal{O}(\log^9 N)$.*

Proof. Our technique is a direct application of Howgrave-Graham's algorithm for approximate integer common divisors [5]. Given two integers $a < b$ and $M = b^\alpha$ for some $\alpha \in [0, 1]$, Howgrave-Graham's algorithm outputs all integers $d > M$ dividing both $a + x_0$ and b for some $|x_0| < X$, in time polynomial in $\log b$, where $X = b^\beta$ and $\beta = \alpha^2$.

Letting $U = e \cdot d - 1$ and $s = p + q - 1$, our goal is to recover s from N and U. Then from s it is straightforward to recover the factorization of N. From $U = 0 \bmod \phi(N)$ and $\phi(N) = (p-1)(q-1) = N - s$, we observe that $N - s$ divides both U and $N - s$. Therefore, one can apply Howgrave-Graham's algorithm with $a := N$, $b := U$, $x_0 := -s$ and $M = N/2$. We have that $\alpha \simeq 1/2$ and $\beta \simeq 1/4$, which enables to recover s and eventually the factorization of N.

In the following, for completeness, we provide the full description of an algorithm for factoring N given (e, d), similar to Howgrave-Graham's algorithm. First, we assume that we are given the high-order bits s_0 of s. More precisely, we let X be some integer, and write $s = s_0 \cdot X + x_0$, where $0 \leq x_0 < X$. The integer s_0 will eventually be recovered by exhaustive search. Moreover, we denote $\phi = \phi(N)$. From $\phi = (p-1) \cdot (q-1) = N - s = N - s_0 \cdot X - x_0$ we obtain the following equations :

$$U = 0 \bmod \phi \tag{3}$$

$$x_0 - N + s_0 \cdot X = 0 \bmod \phi \tag{4}$$

We consider the polynomials :

$$g_{ij}(x) = x^i \cdot (x - N + s_0 \cdot X)^j \cdot U^{m-j}$$

for $0 \leq j \leq m$ and $i = 0$, and for $j = m$ and $1 \leq i \leq k$, where m, k are fixed parameters. From equations (3) and (4), we have that for all previous (i, j) :

$$g_{ij}(x_0) = 0 \bmod \phi^m.$$

For any linear integer combination $h(x)$ of the polynomials $g_{ij}(x)$, we have that $h(x_0) = 0 \bmod \phi^m$. Our goal is then to find a non-zero $h(x)$ with small coefficients. Namely, using the following lemma from [4], if the coefficients of $h(x)$ are sufficiently small, we have that $h(x_0) = 0$ holds over the integers. The integer x_0 can then be recovered using any standard root-finding algorithm; eventually from x_0 one recovers the factorization of N. Given a polynomial $h(x) = \sum h_i x^i$, we denote by $\|h(x)\|$ the Euclidean norm of the vector of its coefficients h_i.

Lemma 1 (Howgrave-Graham). *Let $h(x) \in \mathbb{Z}[x]$ be the sum of at most ω monomials. Suppose that $h(x_0) = 0 \bmod \phi^m$ where $|x_0| \leq X$ and $\|h(xX)\| < \phi^m/\sqrt{\omega}$. Then $h(x_0) = 0$ holds over the integers.*

Proof. We have :

$$\begin{aligned} |h(x_0)| &= \left|\sum h_i x_0^i\right| = \left|\sum h_i X^i \left(\frac{x_0}{X}\right)^i\right| \\ &\leq \sum \left|h_i X^i \left(\frac{x_0}{X}\right)^i\right| \leq \sum |h_i X^i| \\ &\leq \sqrt{\omega}\|h(xX)\| < \phi^m \end{aligned}$$

Since $h(x_0) = 0 \bmod \phi^m$, this gives $h(x_0) = 0$. □

We consider the lattice L spanned by the coefficient vectors of the polynomials $g_{ij}(xX)$. One can see that these coefficient vectors form a triangular basis of a full-rank lattice of dimension $\omega = m + k + 1$ (for an example, see Fig. 1). The determinant of the lattice is then the product of the diagonal entries, which gives :

$$\det L = X^{(m+k)(m+k+1)/2} U^{m(m+1)/2} \tag{5}$$

	1	x	x^2	x^3	x^4	x^5	x^6
$g_{00}(xX)$	U^3						
$g_{01}(xX)$	$*$	U^2X					
$g_{02}(xX)$	$*$	$*$	UX^2				
$g_{03}(xX)$	$*$	$*$	$*$	X^3			
$g_{13}(xX)$		$*$	$*$	$*$	X^4		
$g_{23}(xX)$			$*$	$*$	$*$	X^5	
$g_{33}(xX)$				$*$	$*$	$*$	X^6

Fig. 1. The lattice L of the polynomials $g_{ij}(xX)$ for $k = m = 3$. The symbol '$*$' correspond to non-zero entries whose value is ignored.

Using LLL (Theorem 1), one obtains a non-zero vector b whose norm is guaranteed to satisfy :

$$\|b\| \leq 2^{(\omega-1)/4} \cdot (\det L)^{1/\omega}$$

The vector b is the coefficient vector of some polynomial $h(xX)$ with $\|h(xX)\| = \|b\|$. The polynomial $h(x)$ is then an integer linear combination of the polynomials $g_{ij}(x)$, which implies that $h(x_0) = 0 \bmod \phi^m$. In order to apply Lemma 1, it is therefore sufficient to have that :

$$2^{(\omega-1)/4} \cdot (\det L)^{1/\omega} < \frac{\phi^m}{\sqrt{\omega}}$$

Using the inequalities $\sqrt{\omega} \leq 2^{(\omega-1)/2}$, $\phi > N/2$ and $\omega - 1 = m + k \geq m$, we obtain the following sufficient condition :

$$\det L \leq N^{m \cdot \omega} \cdot 2^{-2 \cdot \omega \cdot (\omega-1)}$$

From equation (5) and inequality $U < N^2$, this gives :

$$X^{(m+k)(m+k+1)/2} \leq N^{m \cdot k} \cdot 2^{-2 \cdot \omega \cdot (\omega - 1)}$$

which gives the following condition for X :

$$X \leq \frac{N^{\gamma(m,k)}}{16}, \quad \gamma(m,k) = \frac{2 \cdot m \cdot k}{(m+k) \cdot (m+k+1)}$$

Our goal is to maximize the bound X on x_0, so that fewer bits must be exhaustively searched. For a fixed m, the function $\gamma(m,k)$ is maximal for $k = m$. The corresponding bound for $k = m$ is then :

$$X \leq \frac{1}{16} \cdot N^{\frac{1}{2} - \frac{1}{4m+2}}. \tag{6}$$

The LLL algorithm is therefore applied on a lattice of dimension $\omega = m+k+1 = 2 \cdot m+1$ and with entries bounded by $B = \mathcal{O}(N^{2m})$. Since the running-time of LLL is polynomial in the lattice dimension and in the size of the entries, given s_0 such that $s = s_0 \cdot X + x_0$ with $0 \leq x_0 < X$, the previous algorithm recovers the factorization of N in time polynomial in $(\log N, m)$.

Finally, taking the greatest integer X satisfying (6), and using $s = p + q - 1 \leq 3N^{\frac{1}{2}}$, we obtain :

$$s_0 \leq \frac{s}{X} \leq 49 \cdot N^{1/(4m+2)}$$

Then, taking $m = \lfloor \log N \rfloor$, we obtain that s_0 is upper-bounded by a constant. The previous algorithm is then run for each possible value of s_0, and the correct s_0 enables to recover the factorization of N. The running time is dominated by the time it takes to run LLL on a lattice of dimension $\omega = 2m + 1$ with entries bounded by $B = \mathcal{O}(N^{2m})$. Since the running time of LLL is bounded by $\mathcal{O}(\omega^6 \log^3 B)$, our algorithm recovers the factorization of N in time $\mathcal{O}(\log^{12} N)$. If one uses the L^2 variant instead of LLL, one obtains a running time of $\mathcal{O}(\log^9 N)$. □

5 Generalization to Unbalanced Prime Factors

The previous algorithm fails when the prime factors p and q are unbalanced, because in this case we have that $s = p + q - 1 \gg \sqrt{N}$, and s is then much greater than the bound on X given by inequality (6).

In this section, we provide an algorithm which extends the result of the previous section to unbalanced prime factors. We use a technique introduced by Durfee and Nguyen in [3], which consists in using two separate variables x and y for the primes p and q, and replacing each occurrence of $x \cdot y$ by N. We note that Howgrave-Graham's algorithm for finding approximate integer common divisors does not seem to apply in this case.

The following theorem shows that the factorization of N given (e,d) becomes easier when the prime factors are imbalanced. Namely, the condition on the product $e \cdot d$ becomes weaker. For example, we obtain that for $p < N^{1/4}$, the modulus N can be factored in polynomial time given (e,d) if $e \cdot d \leq N^{8/3}$ (instead of N^2 for prime factors of equal size).

Theorem 4. *Let β and $0 < \delta \leq 1/2$ be real values, such that $2\beta\delta(1-\delta) \leq 1$. Let $N = p \cdot q$, where p and q are two prime integers such that $p < N^{\delta}$ and $q < 2 \cdot N^{1-\delta}$. Let e, d be such that $e \cdot d = 1 \bmod \phi(N)$, and $1 < e \cdot d \leq N^{\beta}$. Then there is a deterministic algorithm that given (N, e, d) recovers the factorization of N in time $\mathcal{O}(\log^9 N)$.*

Proof. Let $U = ed - 1$ as previously. Our goal is to recover p, q from N and U. We have the following equations :

$$U = 0 \bmod \phi \tag{7}$$

$$p + q - (N + 1) = 0 \bmod \phi \tag{8}$$

Let $m \geq 1$, $a \geq 1$ and $b \geq 0$ be integers. We define the following polynomials $g_{ijk}(x, y)$:

$$g_{ijk}(x, y) = x^i \cdot y^j \cdot U^{m-k} \cdot (x + y - (N + 1))^k$$

$$\begin{cases} i \in \{0, 1\}, & j = 0, \quad k = 0, \ldots, m \\ 1 < i \leq a, & j = 0, \quad k = m \\ i = 0, & 1 \leq j \leq b, \quad k = m \end{cases}$$

In the definition of the polynomials $g_{ijk}(x, y)$, we replace each occurrence of $x \cdot y$ by N; therefore, the polynomials $g_{ijk}(x, y)$ contain only monomials that are powers of x or powers of y. From equations (7) and (8), we obtain that (p, q) is a root of $g_{ijk}(x, y)$ modulo ϕ^m, for all previous (i, j, k) :

$$g_{ijk}(p, q) = 0 \bmod \phi^m$$

Now, we assume that we are given the high-order bits p_0 of p and the high-order bits q_0 of q. More precisely, for some integers X and Y, we write $p = p_0 \cdot X + x_0$ and $q = q_0 \cdot Y + y_0$, with $0 \leq x_0 < X$ and $0 \leq y_0 < Y$. The integers p_0 and q_0 will eventually be recovered by exhaustive search.

We define the translated polynomials :

$$t_{ijk}(x, y) = g_{ijk}(p_0 \cdot X + x, q_0 \cdot Y + y)$$

It is easy to see that for all (i, j, k), we have that (x_0, y_0) is a root of $t_{ijk}(x, y)$ modulo ϕ^m :

$$t_{ijk}(x_0, y_0) = 0 \bmod \phi^m$$

As in the previous algorithm, our goal is to find a non-zero integer linear combination $h(x, y)$ of the polynomials $t_{ijk}(x, y)$, with small coefficients. Then $h(x_0, y_0) = 0 \bmod \phi^m$, and using again Howgrave-Graham's lemma, if the coefficients of $h(x, y)$ are sufficiently small, then $h(x_0, y_0) = 0$ over the integers. Then one can define the polynomial $h'(x) = (p_0 \cdot X + x)^{m+b} \cdot h(x, N/(p_0 \cdot X + x) - q_0 \cdot Y)$. Since $h(x, y)$ is not identically zero and $h(x, y)$ contains only x powers and y powers, the polynomial $h'(x)$ cannot be identically zero. Moreover $h'(x_0) = 0$, which enables to recover x_0 using any standard root-finding algorithm, and eventually the primes p and q. Given a polynomial $h(x, y) = \sum h_{ij} x^i y^j$, we denote by $\|h(x, y)\|$ the Euclidean norm of the vector of its coefficients h_{ij}.

Lemma 2 (Howgrave-Graham). *Let $h(x, y) \in \mathbb{Z}[x, y]$ which is the sum of at most ω monomials. Suppose that $h(x_0, y_0) = 0 \bmod \phi^m$ where $|x_0| \leq X$, $|y_0| \leq Y$ and $\|h(xX, yY)\| < \phi^m / \sqrt{\omega}$. Then $h(x_0, y_0) = 0$ holds over the integers.*

Proof. We have:

$$\begin{aligned}|h(x_0,y_0)| &= \left|\sum h_{ij}x_0^i y_0^i\right| = \left|\sum h_{ij}X^iY^j\left(\frac{x_0}{X}\right)^i\left(\frac{y_0}{Y}\right)^j\right| \\ &\leq \sum\left|h_{ij}X^iY^j\left(\frac{x_0}{X}\right)^i\left(\frac{y_0}{Y}\right)^j\right| \leq \sum\left|h_{ij}X^iY^j\right| \\ &\leq \sqrt{\omega}\|h(xX,yY)\| < \phi^m\end{aligned}$$

Since $h(x_0,y_0) = 0 \bmod \phi^m$, this gives $h(x_0,y_0) = 0$. □

We consider the lattice L spanned by the coefficient vectors of the polynomials $t_{ijk}(xX,yY)$. One can see that these coefficient vectors form a triangular basis of a full-rank lattice of dimension $\omega = 2m+a+b+1$ (for an example, see Fig. 2). The determinant of the lattice is then the product of the diagonal entries, which gives :

$$\det L = X^{(m+a)(m+a+1)/2}Y^{(m+b)(m+b+1)/2}U^{m(m+1)} \tag{9}$$

	1	x	y	x^2	y^2	x^3	y^3	x^4	x^5	y^4
$t_{000}(xX,yY)$	U^3									
$t_{100}(xX,yY)$	$*$	U^3X								
$t_{001}(xX,yY)$	$*$	$*$	U^2Y							
$t_{101}(xX,yY)$	$*$	$*$	$*$	U^2X^2						
$t_{002}(xX,yY)$	$*$	$*$	$*$	$*$	UY^2					
$t_{102}(xX,yY)$	$*$	$*$	$*$	$*$	$*$	UX^3				
$t_{003}(xX,yY)$	$*$	$*$	$*$	$*$	$*$	$*$	Y^3			
$t_{103}(xX,yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	X^4		
$t_{203}(xX,yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	X^5	
$t_{013}(xX,yY)$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$		Y^4

Fig. 2. The lattice L of the polynomials $t_{ijk}(xX,yY)$ for $m = 3$, $a = 2$ and $b = 1$. The symbol '$*$' correspond to non-zero entries whose value is ignored.

As previously, using lattice reduction, one obtains a non-zero polynomial $h(x,y)$ such that:

$$\|h(xX,yY)\| \leq 2^{(\omega-1)/4}\cdot(\det L)^{1/\omega}$$

In order to apply Lemma 2, it is therefore sufficient to have that :

$$2^{(\omega-1)/4}\cdot(\det L)^{1/\omega} < \phi^m/\sqrt{\omega}$$

As in the previous section, using $\sqrt{\omega} \leq 2^{(\omega-1)/2}$, $\phi > N/2$ and $\omega - 1 \geq m$, it is sufficient to have :

$$\det L \leq N^{m\cdot\omega}\cdot 2^{-2\cdot\omega\cdot(\omega-1)} \tag{10}$$

We write $a = \lfloor(u-1)\cdot m - 1\rfloor$ and $b = \lfloor(v-1)\cdot m - 1\rfloor$ for some reals u, v. We obtain that $(m+a)(m+a+1) \leq m^2u^2$ and $(m+b)(m+b+1) \leq m^2v^2$. We write $X = N^{\delta_x}$ and $Y = N^{\delta_y}$ for some reals δ_x, δ_y. From equation (9) and $U \leq N^\beta$ we obtain that :

$$\frac{\log(\det L)}{\log N} \leq m^2 \cdot \left(\delta_x \cdot \frac{u^2}{2} + \delta_y \cdot \frac{v^2}{2} + \beta\right) + \beta \cdot m \tag{11}$$

where log denotes the logarithm in base 2. Moreover, using $m(u+v) - 3 < \omega \leq m(u+v)$, we have :

$$\log\left(N^{m\cdot\omega} \cdot 2^{-2\cdot\omega\cdot(\omega-1)}\right) \geq m\,(m(u+v)-3)\log N - 2m^2(u+v)^2 \tag{12}$$

Therefore, combining inequalities (10), (11) and (12), we obtain the following sufficient condition :

$$u + v - \delta_x\frac{u^2}{2} - \delta_y\frac{v^2}{2} - \beta \geq \frac{\beta+3}{m} + \frac{2}{\log N}(u+v)^2$$

The function $u \to u - \delta_x \cdot u^2/2$ is maximal for $u = 1/\delta_x$, with a maximum equal to $1/(2\delta_x)$. The same holds for the function $v \to v - \delta_y \cdot v^2/2$. Therefore, taking $u = 1/\delta_x$ and $v = 1/\delta_y$, we obtain the sufficient condition :

$$\frac{1}{2\delta_x} + \frac{1}{2\delta_y} - \beta \geq \frac{\beta+3}{m} + \frac{2}{\log N}\left(\frac{1}{\delta_x} + \frac{1}{\delta_y}\right)^2 \tag{13}$$

For $X = N^{\delta_x}$ and $Y = N^{\delta_y}$ satisfying the previous condition and given p_0 and q_0 such that $p = p_0 \cdot X + x_0$ and $q = q_0 \cdot Y + y_0$, the algorithm recovers x_0, y_0 and then p, q in time polynomial in $(m, \log N)$. In the following, we show that p_0 and q_0 can actually be recovered by exhaustive search, while remaining polynomial-time in $\log N$.

Let ε be such that $0 < \varepsilon \leq \delta/2$. We have the following inequalities :

$$\frac{1}{\delta-\varepsilon} = \frac{1}{\delta(1-\frac{\varepsilon}{\delta})} \geq \frac{1}{\delta}\left(1 + \frac{\varepsilon}{\delta}\right) \quad \text{and} \quad \frac{1}{1-\delta-\varepsilon} \geq \frac{1}{1-\delta}\left(1 + \frac{\varepsilon}{1-\delta}\right)$$

From $2\beta\delta(1-\delta) \leq 1$, we obtain :

$$2\beta \leq \frac{1}{\delta(1-\delta)} = \frac{1}{\delta} + \frac{1}{1-\delta}$$

Combining the three previous inequalities, we get :

$$\frac{1}{\delta-\varepsilon} + \frac{1}{1-\delta-\varepsilon} - 2\beta \geq \varepsilon\left(\frac{1}{\delta^2} + \frac{1}{(1-\delta)^2}\right)$$

Therefore, taking $\delta_x = \delta - \varepsilon$ and $\delta_y = 1 - \delta - \varepsilon$, we obtain from (13) the following sufficient condition :

$$\frac{\delta}{2} \geq \varepsilon \geq 2 \cdot \left(\frac{\beta+3}{m} + \frac{2}{\log N}\left(\frac{1}{\delta-\varepsilon} + \frac{1}{1-\delta-\varepsilon}\right)^2\right)\left(\frac{1}{\delta^2} + \frac{1}{(1-\delta)^2}\right)^{-1}$$

Moreover, since $0 < \varepsilon \leq \delta/2$ and $\delta < 1/2$, we have :

$$\frac{1}{\delta-\varepsilon} \leq \frac{2}{\delta} \quad \text{and} \quad \frac{1}{1-\delta-\varepsilon} \leq 4$$

Therefore, this gives the following sufficient condition :

$$\frac{\delta}{2} \geq \varepsilon \geq 2 \cdot \left(\frac{\beta+3}{m} + \frac{2}{\log N} \left(\frac{2}{\delta} + 4 \right)^2 \right) \left(\frac{1}{\delta^2} + \frac{1}{(1-\delta)^2} \right)^{-1}$$

Taking $m = \lfloor \log N \rfloor$, this condition can always be satisfied for large enough $\log N$. Taking the corresponding lower-bound for ε, we obtain $\varepsilon = \mathcal{O}(1/\log N)$, which gives $N^\varepsilon \leq C$ for some constant C. Therefore, we obtain that p_0 and q_0 are upper-bounded by the constants C and $2C$:

$$p_0 \leq \frac{p}{X} \leq N^{\delta - \delta_x} \leq N^\varepsilon \leq C$$

$$q_0 \leq \frac{q}{Y} \leq 2N^{1-\delta-\delta_y} \leq 2N^\varepsilon \leq 2C$$

This shows that p_0 and q_0 can be recovered by exhaustive search while remaining polynomial-time in $\log N$. The total running-time of our algorithm is then dominated by running the lattice reduction algorithm on a lattice basis of dimension $\omega = \mathcal{O}(m)$ and entries bounded by $B = N^{\mathcal{O}(m)}$. Therefore, using LLL, our algorithm recovers the factorization of N in time $\mathcal{O}(\log^{12} N)$. If one uses the L^2 variant instead of LLL, one obtains a running time of $\mathcal{O}(\log^9 N)$. □

6 Practical Experiments

We have implemented the two algorithms of sections 4 and 5, using the LLL implementation of Shoup's NTL library [12]. First, we describe in Table 1 the experiments with prime factors of equal bit-size, with $e \cdot d \simeq N^2$. We assume that we are given the ℓ high-order bits of $s = p + q$; the observed running time for a single execution of LLL is denoted by t. The total running time for factoring N is then estimated as $T \simeq 2^\ell \cdot t$. We obtain that the factorization of N given (e, d) would take a few days for a 512-bit modulus, and a few years for a 1024-bit modulus. This contrasts with Miller's algorithm whose running time is only a fraction of a second for a 1024-bit modulus.

N	bits given	dimension	t	T
512 bits	14 bits	21	70 s	13 days
512 bits	10 bits	29	7 min	5 days
512 bits	9 bits	33	16 min	5 days
1024 bits	26 bits	21	7 min	900 years
1024 bits	19 bits	29	40 min	40 years
1024 bits	17 bits	33	90 min	23 years

Table 1. Bit-size of N, number of bits to be exhaustively searched, lattice dimension, observed running-time for a single LLL-reduction t, and estimated total running-time T, when $e \cdot d \simeq N^2$. The experiments were performed on a 1.6 GHz PC running under Windows 2000/Cygwin.

The experiments with prime factors of unbalanced size and with $e \cdot d \simeq N^2$ are summarized in Table 2. In this case, it was not necessary to know the high-order bits of $s = p + q$, and one recovers the factorization of N after a single application of LLL. The results in Table 2 confirm that the factorization of N is easier when the prime factors are unbalanced.

N	δ	dimension	t
512 bits	0.25	16	2 s
512 bits	0.3	29	2 min
1024 bits	0.25	16	15 s
1024 bits	0.3	29	10 min

Table 2. Bit-size of the RSA modulus N such that $p < N^{\delta}$, lattice dimension, observed running-time for factoring N, when $e \cdot d \simeq N^2$. The experiments were performed on a 1.6 GHz PC running under Windows 2000/Cygwin.

7 Conclusion

We have shown the first *deterministic* polynomial time algorithm that factors an RSA modulus N given the pair of public and secret exponents e and d, provided that $e \cdot d < N^2$. The algorithm is a variant of Coppersmith's technique for finding small roots of univariate modular polynomial equations. We have also provided a generalization to the case of unbalanced prime factors. Finally, we note that the problem of the deterministic polynomial-time equivalence between finding d and factoring N is not entirely solved in this paper, because finding an algorithm for $e \cdot d > N^2$ remains an open problem.

References

1. D. Boneh, G. Durfee and N.A. Howgrave-Graham, *Factoring $n = p^r q$ for large r*, proceedings of Crypto '99, vol. 1666, Lecture Notes in Computer Science.
2. D. Coppersmith, "Small solutions to polynomial equations and low exponent vulnerabilities", Journal of Cryptology, Vol. 10(4), pp. 223–260, 1997.
3. G. Durfee and P. Nguyen "Cryptanalysis of the RSA Schemes with Short Secret Exponent from Asiacrypt'99", Proceedings of Asiacrypt 2000.
4. N. Howgrave-Graham, "Finding small roots of univariate modular equations revisited", Proceedings of Cryptography and Coding, Lecture Notes in Computer Science Vol. 1355, Springer-Verlag, pp. 131–142, 1997
5. N. Howgrave-Graham, "Approximate Integer Common Divisors", Proceedings of CaLC 2001, LNCS 2146, pp. 51-66, 2001. Springer-Verlag.
6. A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients", Mathematische Annalen, Vol. 261, pp. 513–534, 1982
7. A. May, "Computing the RSA Secret Key is Deterministic Polynomial Time Equivalent to Factoring", Proceedings of Crypto 2004, LNCS Vol. 3152, pp. 213-219.
8. G. L. Miller, "Riemann's hypothesis and tests for primality", Seventh Annual ACM Symposium on the Theory of Computing, pp. 234–239, 1975
9. P. Nguyen, D. Stehlé, "Floating-Point LLL Revisited", Proceedings of Eurocrypt 2005, LNCS Vol. 3494, pp. 215–233.
10. P.Q. Nguyen and J. Stern, *The two faces of lattices in cryptology*. Proceedings of CALC '01, LNCS vol. 2146.
11. R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, Vol. 21(2), pp.120–126, 1978

12. V. Shoup, NTL: A Library for doing Number Theory, available online at `http://www.shoup.net/ntl/index.html`

Preuves de sécurité

Les quatres articles suivants concernent le domaine des preuves de sécurités. Les trois premiers permettent d'obtenir des preuves de sécurités améliorées pour certains schémas de chiffrement ou de signature existants. Le quatrième article introduit une nouvelle définition de sécurité pour les fonctions de hachage, et propose des constructions satisfaisant cette définition.

1. **"Optimal Security Proofs for PSS and other Signature Schemes"**, Jean-Sébastien Coron, Proceedings de Eurocrypt 2002.

   Dans cet article, on décrit une nouvelle preuve de sécurité pour le schéma de signature PSS, qui permet de réduire la taille de l'aléa utilisé lors de la génération de la signature et ainsi d'augmenter la bande passante au niveau du message transmis. On introduit aussi dans cet article une technique permettant de montrer qu'une preuve de sécurité pour un schéma de signature est optimale, et nous appliquons cette technique aux schémas FDH et PSS.

2. **"Universal Padding Schemes for RSA"**, Jean-Sébastien Coron, Marc Joye, David Naccache et Pascal Paillier, Proceedings de Crypto 2002.

   Nous montrons dans cet article que l'on peut utiliser le schéma d'encodage de PSS à la fois pour le chiffrement et pour la signature, en utilisant la même clef publique RSA. Cela permet de simplifier l'utilisation pratique de RSA pour le chiffrement et la signature.

3. **"Security Proof for Partial-Domain Hash Signature Schemes"**, Jean-Sébastien Coron, Proceedings de Crypto 2002.

   Nous étudions la sécurité des schémas de signature dans lesquels la taille de la fonction de hachage utilisée lors de l'encodage n'est qu'une fraction de celle du module. Nous montrons que pour $e = 2$ (signatures Rabin), le schéma de signature est sûr dans le modèle de l'oracle aléatoire lorsque la taille du haché est au moins égale aux 2/3 de celle du module. Cela permet de prouver la sécurité des standards de signature ISO 9796-2 et PKCS#1 v1.5, à condition que la taille du haché soit suffisamment grande.

4. **"Merkle-Damgård Revisited : how to Construct a Hash Function"**, Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud et Prashant Puniya, Proceedings de Crypto 2005.

   Dans cet article, nous introduisons une nouvelle notion de sécurité pour les fonctions de hachage, plus forte que la résistance aux collisions. Sous cette définition, une fonction de hachage itérative doit se comporter comme un oracle aléatoire lorsque la primitive permettant de construire la fonction de hachage (fonction de compression ou

block-cipher) est considérée comme idéale. On peut alors utiliser cette fonction dans n'importe quel schéma prouvé sûr dans le modèle de l'oracle aléatoire, et le schéma restera sûr en supposant la fonction de compression (ou le block-cipher) idéale. Nous constatons dans cet article que la construction de Merkle-Damgård, qui est à la base de la plupart des fonctions de hachage (comme par exemple SHA-1) ne permet pas de satisfaire cette définition. Nous proposons donc plusieurs constructions qui permettent de la satisfaire; ces constructions sont faciles à implémenter et très proches de celles utilisées dans la pratique.

Optimal Security Proofs for PSS and other Signature Schemes

Eurocrypt 2002

Jean-Sébastien Coron

Gemplus Card International
34 rue Guynemer
Issy-les-Moulineaux, F-92447, France
coron@clipper.ens.fr

Abstract. The Probabilistic Signature Scheme (PSS) designed by Bellare and Rogaway is a signature scheme provably secure against chosen message attacks in the random oracle model, which security can be tightly related to the security of RSA. We derive a new security proof for PSS in which a much shorter random salt is used to achieve the same security level, namely we show that $\log_2 q_{sig}$ bits suffice, where q_{sig} is the number of signature queries made by the attacker. When PSS is used with message recovery, a better bandwidth is obtained because longer messages can now be recovered. In this paper, we also introduce a new technique for proving that the security proof of a signature scheme is optimal. In particular, we show that the size of the random salt that we have obtained for PSS is optimal: if less than $\log_2 q_{sig}$ bits are used, then PSS is still provably secure but it cannot have a tight security proof. Our technique applies to other signature schemes such as the Full Domain Hash scheme and Gennaro-Halevi-Rabin's scheme, which security proofs are shown to be optimal.

Key-words: Probabilistic Signature Scheme, Provable Security.

1 Introduction

Since the invention of public-key cryptography in the seminal Diffie-Hellman paper [8], significant research endeavors were devoted to the design of practical and provably secure schemes. A proof of security is usually a computational reduction from solving a well established problem to breaking the cryptosystem. Well established problems of cryptographic relevance include factoring large integers, computing discrete logarithms in prime order groups, or extracting roots modulo a composite integer.

For digital signature schemes, the strongest security notion was defined by Goldwasser, Micali and Rivest in [12], as *existential unforgeability under an adaptive chosen message attack*. This notion captures the property that an attacker cannot produce a valid signature, even after obtaining the signature of (polynomially many) messages of his choice.

Goldwasser, Micali and Rivest proposed in [12] a signature scheme based on signature trees which provably meets this definition. The efficiency of the scheme was later improved by Dwork and Naor [9], and Cramer and Damgård [6]. A significant drawback of those signature schemes is that the signature of a message depends on previously signed messages: the signer must thus store information relative to the signatures he generates as time goes by. Gennaro, Halevi and Rabin presented in [11] a new hash-and-sign scheme provably secure against adaptive chosen message attacks which is both state-free and efficient. Its security is based on the strong-RSA assumption. Cramer and Shoup presented in [7] a

signature scheme provably secure against adaptive chosen message attacks, which is also state-free, efficient, and based on the strong-RSA assumption.

The random oracle model, introduced by Bellare and Rogaway in [1], is a theoretical framework allowing to prove the security of hash-and-sign signature schemes. In this model, the hash function is seen as an oracle which outputs a random value for each new query. Bellare and Rogaway defined in [2] the Full Domain Hash (FDH) signature scheme, which is provably secure in the random oracle model assuming that inverting RSA is hard. [2] also introduced the Probabilistic Signature Scheme (PSS), which offers better security guarantees than FDH. Similarly, Pointcheval and Stern [18] proved the security of discrete-log based signature schemes in the random oracle model (see also [15] for a concrete treatment). However, security proofs in the random oracle are not real proofs, since the random oracle is replaced by a well defined hash function in practice; actually, Canetti, Goldreich and Halevi [4] showed that a security proof in the random oracle model does not necessarily imply that a scheme is secure in the real world.

For practical applications of provably secure schemes, the tightness of the security reduction must be taken into account. A security reduction is tight when breaking the signature scheme leads to solving the well established problem with probability close to one. In this case, the signature scheme is almost as secure as the well established problem. On the contrary, if the above probability is too small, the guarantee on the signature scheme will be weak; in which case larger security parameters must be used, thereby decreasing the efficiency of the scheme.

The security reduction of [2] for Full Domain Hash bounds the probability ε of breaking FDH in time t by $(q_{hash} + q_{sig}) \cdot \varepsilon'$ where ε' is the probability of inverting RSA in time t' close to t and where q_{hash} and q_{sig} are the number of hash queries and signature queries performed by the forger. This was later improved in [5] to $\varepsilon \simeq q_{sig} \cdot \varepsilon'$, which is a significant improvement since in practice q_{sig} happens to be much smaller than q_{hash}. However, FDH's security reduction is still not tight, and FDH is still not as secure as inverting RSA.

On the contrary, PSS is almost as secure as inverting RSA ($\varepsilon \simeq \varepsilon'$). Additionally, for PSS to have a tight security proof in [2], the random salt used to generate the signature must be of length at least $k_0 \simeq 2 \cdot \log_2 q_{hash} + \log_2 1/\varepsilon'$, where q_{hash} is the number of hash queries requested by the attacker and ε' the probability of inverting RSA within a given time bound. Taking $q_{hash} = 2^{60}$ and $\varepsilon' = 2^{-60}$ as in [2], we obtain a random salt of size $k_0 = 180$ bits. In this paper, we show that PSS has actually a tight security proof for a random salt as short as $\log_2 q_{sig}$ bits, where q_{sig} is the number of signature queries made by the attacker. For example, for an application in which at most one billion signatures will be generated, $k_0 = 30$ bits of random salt are actually sufficient to guarantee the same level of security as RSA, and taking a longer salt *does not* increase the security level. When PSS is used with message recovery, we obtain a better bandwidth because a larger message can now be recovered when verifying the signature.

Moreover, we show that this size is optimal: if less than $\log_2 q_{sig}$ bits of random salt are used, PSS is still provably secure, but PSS cannot have exactly the same security level as RSA. First, using a new technique, we derive an upper bound for the security of FDH, which shows that the security proof in [5] with $\varepsilon \simeq q_{sig} \cdot \varepsilon'$ is optimal. In other words, it is not possible to further improve the security proof of FDH in order to obtain a security level equivalent to RSA. This answers the open question raised by Bellare and Rogaway

in [2], about the existence of a better security proof for FDH: as opposed to PSS, FDH *cannot* be proven as secure as inverting RSA. The technique also applies to other signature schemes such as Gennaro-Halevi-Rabin's scheme [11] and Paillier's signature scheme [16]. To our knowledge, this is the first result concerning optimal security proofs. Then, using the upper bound for the security of FDH, we show that our size k_0 for the random salt in PSS is optimal: if less than $\log_2 q_{sig}$ bits are used, no security proof for PSS can be tight.

2 Definitions

In this section we briefly present some notations and definitions used throughout the paper. We start by recalling the definition of a signature scheme.

Definition 1 (signature scheme). *A signature scheme* $(\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ *is defined as follows:*

- The key generation algorithm `Gen` *is a probabilistic algorithm which given* 1^k*, outputs a pair of matching public and private keys,* (pk, sk)*.*

- The signing algorithm `Sign` *takes the message* M *to be signed, the public key* pk *and the private key* sk*, and returns a signature* $x = \mathtt{Sign}_{pk,sk}(M)$*. The signing algorithm may be probabilistic.*

- The verification algorithm `Verify` *takes a message* M*, a candidate signature* x' *and* pk*. It returns a bit* $\mathtt{Verify}_{pk}(M, x')$*, equal to one if the signature is accepted, and zero otherwise. We require that if* $x \leftarrow \mathtt{Sign}_{pk,sk}(M)$*, then* $\mathtt{Verify}_{pk}(M, x) = 1$*.*

In the previously introduced *existential unforgeability under an adaptive chosen message attack* scenario, the forger can dynamically obtain signatures of messages of his choice and attempts to output a valid forgery. A *valid forgery* is a message/signature pair (M, x) such that $\mathtt{Verify}_{pk}(M, x) = 1$ whereas the signature of M was never requested by the forger.

A significant line of research for proving the security of signature schemes is the previously introduced random oracle model, where resistance against adaptive chosen message attacks is defined as follows [1]:

Definition 2. *A forger* $\mathcal{F}$ *is said to* $(t, q_{hash}, q_{sig}, \varepsilon)$*-break the signature scheme* (`Gen`, `Sign`, `Verify`) *if after at most* $q_{hash}(k)$ *queries to the hash oracle,* $q_{sig}(k)$ *signatures queries and* $t(k)$ *processing time, it outputs a valid forgery with probability at least* $\varepsilon(k)$ *for all* $k \in \mathbb{N}$*.*

and quite naturally:

Definition 3. *A signature scheme* (`Gen`, `Sign`, `Verify`) *is* $(t, q_{sig}, q_{hash}, \varepsilon)$*-secure if there is no forger who* $(t, q_{hash}, q_{sig}, \varepsilon)$*-breaks the scheme.*

The RSA cryptosystem, invented by Rivest, Shamir and Adleman [19], is the most widely used cryptosystem today:

Definition 4 (The RSA cryptosystem). *The RSA cryptosystem is a family of trapdoor permutations, specified by:*

- The RSA generator $\mathcal{RSA}$*, which on input* 1^k*, randomly selects two distinct* $k/2$*-bit primes* p *and* q *and computes the modulus* $N = p \cdot q$*. It randomly picks an encryption*

exponent $e \in \mathbb{Z}^*_{\phi(N)}$ *and computes the corresponding decryption exponent* d *such that* $e \cdot d = 1 \bmod \phi(N)$. *The generator returns* (N, e, d).

- *The encryption function* $f : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ *defined by* $f(x) = x^e \bmod N$.
- *The decryption function* $f^{-1} : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ *defined by* $f^{-1}(y) = y^d \bmod N$.

FDH was the first practical and provably secure signature scheme based on RSA. It is defined as follows: the key generation algorithm, on input 1^k, runs $\mathcal{RSA}(1^k)$ to obtain (N, e, d). It outputs (pk, sk), where the public key pk is (N, e) and the private key sk is (N, d). The signing and verifying algorithms use a hash function $H : \{0,1\}^* \to \mathbb{Z}^*_N$ which maps bit strings of arbitrary length to the set of invertible integers modulo N.

$\texttt{SignFDH}_{N,d}(M)$	$\texttt{VerifyFDH}_{N,e}(M, x)$
$y \leftarrow H(M)$	$y \leftarrow x^e \bmod N$
return $y^d \bmod N$	if $y = H(M)$ then return 1 else return 0.

FDH is provably secure in the random oracle model, assuming that inverting RSA is hard. An *inverting algorithm* $\mathcal{I}$ for RSA gets as input (N, e, y) and tries to find $y^d \bmod N$. Its success probability is the probability to output $y^d \bmod N$ when (N, e, d) are obtained by running $\mathcal{RSA}(1^k)$ and y is set to $x^e \bmod N$ for some x chosen at random in $\mathbb{Z}^*_N$.

Definition 5. *An inverting algorithm* $\mathcal{I}$ *is said to* (t, ε)*-break RSA if after at most* $t(k)$ *processing time its success probability is at least* $\varepsilon(k)$ *for all* $k \in \mathbb{N}$.

Definition 6. *RSA is said to be* (t, ε)*-secure if there is no inverter which* (t, ε)*-breaks RSA.*

The following theorem [5] proves the security of FDH in the random oracle model. We include the proof in appendix A for further reference in the paper.

Theorem 1. *Assuming that RSA is* (t_I, ε_I)*-secure, FDH is* $(t_F, q_{hash}, q_{sig}, \varepsilon_F)$*-secure, with:*

$$t_I = t_F + (q_{hash} + q_{sig} + 1) \cdot \mathcal{O}(k^3) \tag{1}$$

$$\varepsilon_I = \frac{\varepsilon_F}{q_{sig}} \cdot \left(1 - \frac{1}{q_{sig} + 1}\right)^{q_{sig}+1} \tag{2}$$

The technique described in [5] can be used to obtain an improved security proof for Gennaro-Halevi-Rabin's signature scheme [11] in the random oracle model and for Paillier's signature scheme [16]. From a forger which outputs a forgery with probability ε_F, the reduction succeeds in solving the hard problem with probability roughly ε_F / q_{sig}, in approximately the same time bound.

For example, if we assume that, for a given security parameter k, the probability of inverting RSA is less than 2^{-60} for a given time bound t, and if the forger is allowed to make at most 2^{60} hash queries and 2^{30} signature queries, then the probability of breaking FDH is less than 2^{-28} for a time bound close to t.

The security reduction of FDH is not tight: the probability ε_F of breaking FDH is smaller than roughly $q_{sig} \cdot \varepsilon_I$ where ε_I is the probability of inverting RSA, whereas the security reduction of PSS is tight: the probability of breaking PSS is almost the same as the probability of inverting RSA ($\varepsilon_F \simeq \varepsilon_I$).

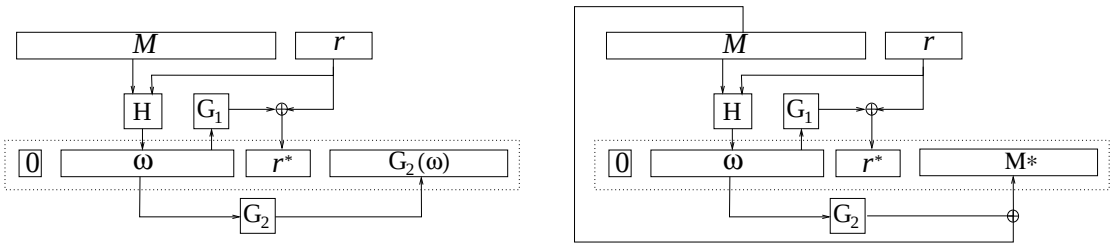

Fig. 1. PSS (left) and PSS-R (right)

3 New Security Proof for PSS

Several standards include PSS [2], among these are IEEE P1363a [13], a revision of ISO/IEC 9796-2, and the upcoming PKCS#1 v2.1 [17]. The signature scheme PSS is parameterized by the integers k, k_0 and k_1. The key generation is identical to FDH. The signing and verifying algorithms use two hash functions $H : \{0,1\}^* \to \{0,1\}^{k_1}$ and $G : \{0,1\}^{k_1} \to \{0,1\}^{k-k_1-1}$. Let G_1 be the function which on input $\omega \in \{0,1\}^{k_1}$ returns the first k_0 bits of $G(\omega)$, whereas G_2 is the function returning the remaining $k-k_0-k_1-1$ bits of $G(\omega)$. A random *salt* r of k_0 bits is concatenated to the message M before hashing it. The scheme is illustrated in figure 1. In this section we obtain a better security proof for PSS, in which a shorter random salt is used to generate the signature.

`SignPSS`(M) :	`VerifyPSS`(M,x) :
$r \overset{R}{\leftarrow} \{0,1\}^{k_0}$	$y \leftarrow x^e \bmod N$
$\omega \leftarrow H(M\|r)$	Break up y as $b\|\omega\|r^*\|\gamma$
$r^* \leftarrow G_1(\omega) \oplus r$	Let $r \leftarrow r^* \oplus G_1(\omega)$
$y \leftarrow 0\|\omega\|r^*\|G_2(\omega)$	if $H(M\|r) = \omega$ and $G_2(\omega) = \gamma$ and $b = 1$
return $y^d \bmod N$	then return 1 else return 0

The following theorem [2] proves the security of PSS in the random oracle model:

Theorem 2. *Assuming that RSA is (t',ε')-secure, the signature scheme* `PSS`$[k_0,k_1]$ *is* $(t, q_{sig}, q_{hash}, \varepsilon)$*-secure, where :*

$$t = t' - (q_{hash} + q_{sig} + 1) \cdot k_0 \cdot \mathcal{O}(k^3) \tag{3}$$

$$\varepsilon = \varepsilon' + 3 \cdot (q_{sig} + q_{hash})^2 \cdot \left(2^{-k_0} + 2^{-k_1}\right) \tag{4}$$

Theorem 2 shows that for PSS to be as secure as RSA (*i.e.* $\varepsilon' \simeq \varepsilon$), it must be the case that $(q_{sig} + q_{hash})^2 \cdot \left(2^{-k_0} + 2^{-k_1}\right) < \varepsilon'$, which gives $k_0 \geq k_{min}$ and $k_1 \geq k_{min}$, where:

$$k_{min} = 2 \cdot \log_2(q_{hash} + q_{sig}) + \log_2 \frac{1}{\varepsilon'} \tag{5}$$

Taking $q_{hash} = 2^{60}$, $q_{sig} = 2^{30}$ and $\varepsilon' = 2^{-60}$ as in [2], we obtain that k_0 and k_1 must be greater than $k_{min} = 180$ bits.

The following theorem shows that PSS can be proven as secure as RSA for a much shorter random salt, namely $k_0 = \log_2 q_{sig}$ bits, which for $q_{sig} = 2^{30}$ gives $k_0 = 30$ bits. The minimum value for k_1 remains unchanged.

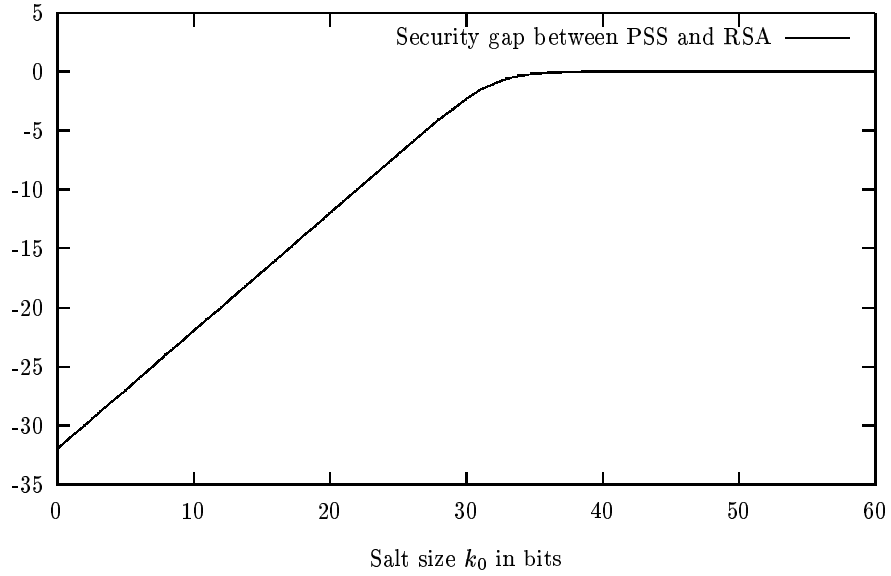

Fig. 2. Security gap between PSS and RSA: $\log_2 \varepsilon'/\varepsilon$ as a function of the salt size k_0 for $q_{sig} = 2^{30}$ signature queries.

Theorem 3. *Assuming that RSA is* (t', ε')*-secure, the signature scheme* $\mathtt{PSS}[k_0, k_1]$ *is* $(t, q_{sig}, q_{hash}, \varepsilon)$*-secure, where :*

$$t = t' - (q_{hash} + q_{sig}) \cdot k_1 \cdot \mathcal{O}(k^3) \tag{6}$$

$$\varepsilon = \varepsilon' \cdot \left(1 + 6 \cdot q_{sig} \cdot 2^{-k_0}\right) + 2 \cdot (q_{hash} + q_{sig})^2 \cdot 2^{-k_1} \tag{7}$$

The proof is given in appendix B. The difference with the previous security proof is the following: in [2], a new random salt r is randomly generated for each signature query, and if r has appeared before, the inverter stops and has failed. Since at most $q_{hash} + q_{sig}$ random salts can appear during the reduction, the inverter stops after a given signature query with probability less than $(q_{hash} + q_{sig}) \cdot 2^{-k_0}$. There are at most q_{sig} signature queries, so this gives an error probability of:

$$q_{sig} \cdot (q_{hash} + q_{sig}) \cdot 2^{k_0}$$

which accounts for the term $(q_{hash} + q_{sig})^2 \cdot 2^{-k_0}$ in equation (4). On the contrary, in our new security proof, we generate for each new message M_i a list of q_{sig} random salts. Those random salts are then used to answer the signature queries for M_i, so there is no error probability when answering the signature queries.

3.1 Discussion

Theorem 3 shows that PSS is actually provably secure for any size k_0 of the random salt. In figure 2 we plot $\log_2 \varepsilon'/\varepsilon$ as a function of the size k_0 of the salt, which depicts the relative security of PSS compared to RSA, for $q_{sig} = 2^{30}$ and $k_1 > k_{min}$. For $k_0 = 0$, we reach the security level of FDH, where approximately $\log_2 q_{sig}$ bits of security are lost compared to

RSA. For k_0 comprised between zero and $\log_2 q_{sig}$, we gain one bit of security when k_0 increases by one bit. And for k_0 greater than $\log_2 q_{sig}$, the security level of PSS is almost the same as inverting RSA. This shows that PSS has a tight security proof as soon as the salt size reaches $\log_2 q_{sig}$, and using larger salts does not further improve security. For the signer, q_{sig} represents the maximal number of signatures which can be generated for a given public-key. For example, for an application in which at most one billion signatures will be generated, $k_0 = 30$ bits of random salt are actually sufficient to guarantee the same level of security as RSA, and taking a larger salt does not increase the security level.

More precisely, taking $k_0 = \log_2 q_{sig}$ and $k_1 = k_{min}$ where k_{min} is given by (5), we obtain that the probability of breaking PSS in time less than t, is less than $\varepsilon = 9 \cdot \varepsilon'$, where ε' is the probability of inverting RSA in time close to t. Therefore with those parameters PSS is almost as secure as inverting RSA[1]. Taking $q_{hash} = 2^{60}$, $q_{sig} = 2^{30}$ and $\varepsilon' = 2^{-60}$ for a 1024-bit modulus as in [2], we can take $k_1 = k_{min} = 180$ bits and $k_0 = \log_2 q_{sig} = 30$ bits.

PSS-R is a variant of PSS which provides message recovery; the scheme is illustrated in figure 1. The goal is to save on the bandwidth: instead of transmitting the message separately, the message is recovered when verifying the signature. The security proof for PSS-R is almost identical to the security proof of PSS, and PSS-R achieves the same security level as PSS. Consequently, using the same parameters as for PSS with a 1024-bits RSA modulus, 813 bits of message can now be recovered when verifying the signature (instead of 663 bits with the previous security proof).

4 Optimal Security Proof for FDH

In section 2 we have seen that the security proof of theorem 1 for FDH is still not tight: the probability ε_F of breaking FDH is smaller than roughly $q_{sig} \cdot \varepsilon_I$ where ε_I is the probability of inverting RSA, whereas the security reduction of PSS is tight: the probability of breaking PSS is almost the same as the probability of inverting RSA ($\varepsilon_F \simeq \varepsilon_I$). An interesting question is whether it is possible to obtain a better security bound for FDH. In particular, is it possible to show that FDH is as secure as inverting RSA ?

In this section we show that the security proof of theorem 1 for FDH is optimal, *i.e.* there is no better reduction from inverting RSA to breaking FDH, and one cannot avoid loosing the q_{sig} factor in the probability bound. A possible direction would be to demonstrate an attack against FDH which would not apply to inverting RSA. More precisely, if we could prove that the best possible attack against FDH is q_{sig} times faster than the best possible attack against RSA, this would show that FDH is indeed less secure than RSA and that the previous security proof for FDH is optimal. But actually we don't know any attack on FDH, faster than factoring N.

Instead, in order to show that there is no better reduction from inverting RSA to breaking FDH, we will use a similar approach as Boneh and Venkatesan in [3] for disproving the equivalence between inverting low-exponent RSA and factoring. They show that any efficient algebraic reduction from factoring to inverting low-exponent RSA can be converted into an efficient factoring algorithm. Such reduction is an algorithm $\mathcal{A}$ which factors N using an e-th root oracle for N. They show how to convert $\mathcal{A}$ into an algorithm $\mathcal{B}$ that

[1] The factor 9 is not relevant here, because it represents less than 4 bits of security. To obtain $\varepsilon' \simeq \varepsilon$, we can take $k_0 = \log_2 q_{sig} + 8$ and $k_1 = k_{min} + 8$, which gives $\varepsilon' = 1.04 \cdot \varepsilon$.

factors integers without using the e-th root oracle. Thus, unless factoring is easy, inverting low-exponent RSA cannot be equivalent to factoring under algebraic reductions.

Similarly, we show that any better reduction from inverting RSA to breaking FDH can be converted into an efficient RSA inverting algorithm. Such reduction is an algorithm $\mathcal{R}$ which uses a forger as an oracle in order to invert RSA. We show how to convert $\mathcal{R}$ into an algorithm $\mathcal{I}$ which inverts RSA without using the oracle forger. Consequently, if inverting RSA is hard, there is no such better reduction for FDH, and the reduction of theorem 1 must be optimal.

Our technique is the following. Recall that resistance against adaptive chosen message attacks is considered, so the forger is allowed to make signature queries for messages of its choice, which must be answered by the reduction $\mathcal{R}$. Eventually the forger outputs a forgery, and the reduction must invert RSA. Therefore we first ask the reduction to sign a message M and receive its signature s, then we rewind the reduction to the state in which it was before the signature query, and we send s as a forgery for M. This is a true forgery for the reduction, because after the rewind there was no signature query for M, so eventually the reduction inverts RSA. Consequently, we have constructed from $\mathcal{R}$ an algorithm $\mathcal{I}$ which inverts RSA without using any forger. Actually, this technique allows to simulate a forger with respect to $\mathcal{R}$, without being able to break FDH. However, the simulation is not perfect, because it outputs a forgery only for messages which can be signed by the reduction, whereas a real forger outputs the forgery of a message which the reduction may or may not be able to sign.

We quantify the efficiency of a reduction by giving the probability that the reduction inverts RSA using a forger that $(t_F, q_{hash}, q_{sig}, \varepsilon_F)$-breaks the signature scheme, within an additional running time of t_R:

Definition 7. *We say that a reduction algorithm $\mathcal{R}$ $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces inverting RSA to breaking FDH if upon input (N, e, y) and after running any forger that $(t_F, q_{hash}, q_{sig}, \varepsilon_F)$-breaks FDH, the reduction outputs $y^d \bmod N$ with probability greater than ε_R, within an additional running time of t_R.*

In the above definition, t_R is the running time of the reduction algorithm only and does not include the running time of the forger. Eventually, the time needed to invert RSA is $t_F + t_R$, where t_F is the running time of the forger. For example, the reduction of theorem 1 for FDH $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces inverting RSA to breaking FDH with $t_R(k) = (q_{hash} + q_{sig}) \cdot \mathcal{O}(k^3)$ and $\varepsilon_R = \varepsilon_F/(4 \cdot q_{sig})$.

The following theorem shows that from any such reduction $\mathcal{R}$ we can invert RSA with probability greater than roughly $\varepsilon_R - \varepsilon_F/q_{sig}$, in roughly the same time bound. The term ε_F/q_{sig} is due to the fact that our simulation of a forger is not perfect. This also corresponds to the success probability of the reduction in theorem 1. This means that if the success probability ε_R of the reduction is greater than ε_F/q_{sig}, we obtain an algorithm which inverts RSA without using the forger. Therefore, if inverting RSA is hard, the success probability of the reduction cannot be greater than roughly ε_F/q_{sig}, and the reduction of theorem 1 must be optimal. The proof is given in appendix C.

Theorem 4. *Let $\mathcal{R}$ be a reduction which $(t_R, q_{hash}, q_{sig}, \varepsilon_R, \varepsilon_F)$-reduces inverting RSA to breaking FDH. $\mathcal{R}$ runs the forger only once. From $\mathcal{R}$ we can construct an algorithm which (t_I, ε_I)-inverts RSA, with:*

$$t_I = 2 \cdot t_R \tag{8}$$

$$\varepsilon_I = \varepsilon_R - \varepsilon_F \cdot \frac{\exp(-1)}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1} \tag{9}$$

4.1 Discussion

Theorem 4 shows that from any reduction $\mathcal{R}$ which inverts RSA with probability ε_R when interacting with a forger which outputs a forgery with probability ε_F, we can invert RSA with probability roughly $\varepsilon_R - \varepsilon_F/q_{sig}$, in roughly the same time bound, without using a forger. For simplicity, we omit here the factors $\exp(-1)$ and $(1-q_{sig}/q_{hash})$ in equation (9). Moreover we consider a forger which makes q_{sig} signature queries, and with probability $\varepsilon_F = 1$ outputs a forgery[2].

Theorem 4 implies that from a polynomial time reduction $\mathcal{R}$ which succeeds with probability ε_R when interacting with this forger, we obtain a polynomial time RSA inverter $\mathcal{I}$ which succeeds with probability $\varepsilon_I = \varepsilon_R - 1/q_{sig}$, without using the forger. If inverting RSA is hard, the success probability ε_I of the polynomial time inverter must be negligible. Consequently, the success probability ε_R of the reduction must be less than $1/q_{sig} + \texttt{negl}$. This shows that from a forger which outputs a forgery with probability one, a polynomial time reduction cannot succeed with probability greater than $1/q_{sig}+\texttt{negl}$. On the contrary, a tight security reduction would invert RSA with probability close to one. Here we cannot avoid the q_{sig} factor in the security proof: the security level of FDH cannot be proven equivalent to RSA, and the security proof of theorem 1 for FDH is optimal.

5 Extension to any Signature Scheme with Unique Signature

We have introduced a new technique which enables to simulate a forger with respect to a reduction. It consists in making a signature query for a message M, rewinding the reduction, then sending the signature of M as a forgery. Actually, this technique stretches beyond FDH and can be generalized and applied to any signature scheme. However, the technique works only for signature schemes in which each message has a unique signature, because otherwise the forger cannot be simulated. Namely if M has many possible signatures, our simulation sends as a forgery for M a signature s that was received from $\mathcal{R}$, whereas a real forger has no information about s (since it has not queried M for signature to $\mathcal{R}$) and can output any signature $s' \neq s$ for M. For signature schemes with unique signature, our technique shows that the reduction cannot succeed with probability greater than roughly ε_F/q_{sig}, using a forger which outputs a forgery with probability ε_F. Signature schemes with unique signature include FDH, Gennaro-Halevi-Rabin's signature scheme and Paillier's signature scheme. Note that PSS is not a signature scheme with unique signature (except if $k_0 = 0$).

However, we have so far considered reductions running a forger only once. If the reduction of theorem 1 for FDH runs the forger r times, its success probability will be roughly $r \cdot \varepsilon_F/q_{sig}$, and the total running time will be roughly r times the running time of the forger. But there might be a better reduction which would yield a better time/probability

[2] Such forger can be constructed by first factoring the modulus N, then computing a forgery using the factorisation of N.

trade-off. For example, a reduction for FDH could succeed in inverting RSA with probability $\varepsilon_R \simeq \varepsilon_F$ when running a forger only twice. In this case, FDH would be almost as secure as inverting RSA. Additionally, the reduction might rewind the forger with different inputs, as for proof-of-knowledge based signature schemes [15, 18].

The following theorem shows that there is no better time/probability trade-off: for a hash-and-sign signature scheme with unique signature, a reduction allowed to run or rewind a forger at most r times cannot succeed with probability greater than roughly $r \cdot \varepsilon_F / q_{sig}$. The definitions are in appendix E and the proof of the theorem is given in appendix F.

Theorem 5. *Let $\mathcal{R}$ be a reduction which $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces solving a problem Π to breaking a hash-and-sign signature scheme with unique signature. $\mathcal{R}$ is allowed to run or rewind a forger at most r times. From $\mathcal{R}$ we can construct an algorithm which (t_A, ε_A)-solves Π, with:*

$$t_A = (r+1) \cdot t_R \tag{10}$$

$$\varepsilon_A = \varepsilon_R - \varepsilon_F \cdot \frac{\exp(-1) \cdot r}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1} \tag{11}$$

6 Security Proofs for Signature Schemes in the Standard Model

The same technique can be applied to security reductions in the standard model, and we obtain the same upper bound in $1/q_{sig}$ for signature schemes with unique signature. The definitions of security against adaptive chosen message attacks are analogous in the standard model and can be found in appendix G.

The following theorem is analogous to theorem 5. It proves that for any signature scheme with unique signature, assuming the hardness of a given problem Π, any security reduction running or rewinding a forger at most r times cannot be tighter than roughly $r \cdot \varepsilon_F / q_{sig}$. Namely a better reduction can be converted into an algorithm for solving Π, in approximately the same time bound. The proof is similar to the proof of theorem 5 and is given in appendix H

Theorem 6. *Let $\mathcal{R}$ be a reduction which $(t_R, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces solving Π to breaking a signature scheme with unique signature. $\mathcal{R}$ can run or rewind the forger at most r times. Assume that the size of the message space is at least 2^ℓ. From $\mathcal{R}$ we can construct an algorithm which (t_A, ε_A)-solves Π, with:*

$$t_A = (r+1) \cdot t_R \tag{12}$$

$$\varepsilon_A = \varepsilon_R - \varepsilon_F \cdot \frac{\exp(-1) \cdot r}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{2^\ell}\right)^{-1} \tag{13}$$

Gennaro-Halevi-Rabin's signature scheme has a tight ($\varepsilon_F \simeq \varepsilon_R$) security reduction in the standard model, but the above theorem does not apply here because the reduction of [11] requires that a message has many possible signatures. This is also the case for the Cramer-Shoup signature scheme [7]. However, we show in appendix I that the above bound in $1/q_{sig}$ is reached for a variant of Gennaro-Halevi-Rabin's scheme with unique signature, provably secure in the standard model, but for short messages only (say, less than 40 bits). We do not know if there exists a *practical* signature scheme with unique signature, provably secure in the standard model and reaching the above bound.

7 Optimal Security Proof for PSS

In section 3 we have seen that $k_0 = \log_2 q_{sig}$ bits of random salt are sufficient for PSS to have a security level equivalent to RSA, and taking a larger salt does not further improve the security. An interesting question is that of knowing whether this size is optimal or not. Note that for k_1, the output size of the hash function H, the minimum value k_{min} given by equation (5) is clearly optimal, because an attacker making q_{hash} hash queries can find a collision $H(M||0) = H(M'||0)$ with probability roughly $(q_{hash})^2 \cdot 2^{-k_1}/2$ and then forge the signature of M' using the signature of M. However, there might be a better security proof for PSS which would be tight for a shorter size k_0 of the random salt. Actually, if this size is equal to zero, the scheme becomes with unique signature, and we know from section 5 that one must loose $\log_2 q_{sig}$ bits of security compared to the security of RSA. So it seems natural to think that we need at least $\log_2 q_{sig}$ bits of random to make PSS as secure as RSA, because normally we should gain at most one bit of security for each added bit of random salt.

In this section, we show that this is indeed the case: if a shorter random salt is used, the security level of PSS cannot be proven equivalent to RSA. Our technique described in section 4 does not apply directly because PSS is not a signature scheme with unique signature. We extend our technique to PSS using the following method.

We consider PSS in which the random salt is fixed to 0^{k_0}, and we denote this signature scheme $\texttt{PSS0}[k_0, k_1]$. Consequently, $\texttt{PSS0}[k_0, k_1]$ is a signature scheme with unique signature. First, we show how to convert a forger for $\texttt{PSS0}[k_0, k_1]$ into a forger for $\texttt{PSS}[k_0, k_1]$. A reduction $\mathcal{R}$ from inverting RSA to breaking $\texttt{PSS}[k_0, k_1]$ uses a forger for $\texttt{PSS}[k_0, k_1]$ in order to invert RSA. Consequently, from a forger for $\texttt{PSS0}[k_0, k_1]$, we can invert RSA using the reduction $\mathcal{R}$. This means that from $\mathcal{R}$ we can construct a reduction $\mathcal{R}_0$ from inverting RSA to breaking $\texttt{PSS0}[k_0, k_1]$. Since $\texttt{PSS0}[k_0, k_1]$ is a signature scheme with unique signature, theorem 5 gives an upper bound for the success probability of $\mathcal{R}_0$, from which we derive an upper bound for the success probability of $\mathcal{R}$. More precisely, we show that from a reduction $\mathcal{R}$ which inverts RSA in time t_R with probability ε_R when running at most r times a forger which breaks $\texttt{PSS}[k_0, k_1]$ with probability ε_F, one can invert RSA without using the forger, with probability $\varepsilon_I = \varepsilon_R - r \cdot \varepsilon_F \cdot 2^{k_0+2}/q_{sig}$, in time $t_I = (r+1) \cdot t_R$. This shows that the size k_0 of the random salt must be at least $\log_2 q_{sig}$ for $\texttt{PSS}[k_0, k_1]$ to have a security level equivalent to RSA, and so our security proof of section 3 is optimal.

Theorem 7. *Let $\mathcal{R}$ a reduction which $(t, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces inverting RSA to breaking $\texttt{PSS}[k_0, k_1]$, with $q_{hash} \geq 2 \cdot q_{sig}$. The reduction can run or rewind the forger at most r times. From $\mathcal{R}$ we can construct an inverting algorithm for RSA which (t_I, ε_I)-inverts RSA, with:*

$$t_I = (r+1) \cdot (t_R + q_{sig} \cdot \mathcal{O}(k)) \tag{14}$$

$$\varepsilon_I = \varepsilon_R - r \cdot \varepsilon_F \cdot \frac{2^{k_0+2}}{q_{sig}} \tag{15}$$

Proof. The proof is given in appendix J.

7.1 Discussion

Let consider as in section 4.1 a forger for $\mathtt{PSS}[k_0, k_1]$ which makes q_{sig} signature queries and outputs a forgery with probability $\varepsilon_F = 1/2$. Then, from a polynomial time reduction $\mathcal{R}$ which succeeds with probability ε_R when running once this forger, we obtain a polynomial time inverter which succeeds with probability $\varepsilon_I = \varepsilon_R - 2^{k_0+1}/q_{sig}$, without using the forger. If inverting RSA is hard, the success probability ε_I of the polynomial time inverter must be negligible, and therefore the success probability ε_R of the reduction must be less than $2^{k_0+1}/q_{sig}+\mathtt{negl}$. Consequently, in order to have a tight security reduction ($\varepsilon_R \simeq \varepsilon_R$), we must have $k_0 \simeq \log_2 q_{sig}$. The reduction of theorem 3 is consequently optimal.

8 Conclusion

We have described a new technique for analyzing the security proofs of signature schemes. The technique is both general and very simple and allows to derive upper bounds for security reductions using a forger as a black box, both in the random oracle model and in the standard model, for signature schemes with unique signature. We have also obtained a new criterion for a security reduction to be optimal, which may be of independent interest: we say that a security reduction is optimal if from a better reduction one can solve a difficult problem, such as inverting RSA. Our technique enables to show that the Full Domain Hash scheme, Gennaro-Halevi-Rabin's scheme and Paillier's signature scheme have an optimal security reduction in that sense. In other words, we have a matching lower and upper bound for the security reduction of those signature schemes: one cannot do better than losing a factor of q_{sig} in the security reduction.

Moreover, we have described a better security proof for PSS, in which a much shorter random salt is sufficient to achieve the same security level. This is of practical interest, since when PSS is used with message recovery, a better bandwidth is obtained because larger messages can be embedded inside the signature. Eventually, we have shown that this security proof for PSS is optimal: if a smaller random salt is used, PSS remains provably secure, but it cannot have the same level of security as RSA.

Acknowledgements :
I would like to thank Burt Kaliski, Jacques Stern and David Pointcheval for helpful discussions and the anonymous referees for their comments.

References

1. M. Bellare and P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Commmunications Security, ACM, 1993.
2. M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*. Proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, 1996, pp. 399-416.
3. D. Boneh and R. Venkatesan, *Breaking RSA may not be equivalent to factoring*. Proceedings of Eurocrypt' 98, LNCS vol. 1403, Springer-Verlag, 1998, pp. 59–71.

4. R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology, revisited*, STOC' 98, ACM, 1998.
5. J.S. Coron, *On the exact security of Full Domain Hash*, Proceedings of Crypto'2000, LNCS vol. 1880, Springer-Verlag, 2000, pp. 229-235.
6. R. Cramer and I. Damgård, *New generation of secure and practical RSA-based signatures*, Proceedings of Crypto'96, LNCS vol. 1109, Springer-Verlag, 1996, pp. 173-185.
7. R. Cramer and V. Shoup, *Signature schemes based on the Strong RSA Assumption*, May 9, 2000, revision of the extended abstract in Proc. 6th ACM Conf. on Computer and Communications Security, 1999; To appear, ACM Transactions on Information and System Security (ACM TISSEC). Available at `http://www.shoup.net/`
8. W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, IT-22, 6, pp. 644-654, 1976.
9. C. Dwork and M. Naor, *An efficient existentially unforgeable signature scheme and its applications*, In J. of Cryptology, 11 (3), Summer 1998, pp. 187-208.
10. FIPS 186, *Digital signature standard*, Federal Information Processing Standards Publication 186, U.S. Department of Commerce/NIST, 1994.
11. R. Gennaro, S. Halevi and T. Rabin, *Secure hash-and-sign signatures without the random oracle*, proceedings of Eurocrypt '99, LNCS vol. 1592, Springer-Verlag, 1999, pp. 123-139.
12. S. Goldwasser, S. Micali and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of computing, 17(2), pp. 281-308, April 1988.
13. IEEE P1363a, *Standard Specifications For Public Key Cryptography: Additional Techniques*, available at `http://www.manta.ieee.org/groups/1363`
14. A. Lenstra and H. Lenstra (eds.), *The development of the number field sieve*, Lecture Notes in Mathematics, vol 1554, Springer-Verlag, 1993.
15. K. Ohta and T. Okamoto, *On concrete security treatment of signatures derived from identification*. Prooceedings of Crypto '98, Lecture Notes in Computer Science vol. 1462, Springer-Verlag, 1998, pp. 354-369.
16. P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes*. Proceedings of Eurocrypt'99, LNCS vol. 1592, Springer-Verlag, 1999, pp. 223-238.
17. PKCS #1 v2.1, *RSA Cryptography Standard (draft)*, available online at `www.rsasecurity.com/rsalabs/pkcs`.
18. D. Pointcheval and J. Stern, *Security proofs for signature schemes*. Proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, pp. 387-398.
19. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, CACM 21, 1978.

A Proof of Theorem 1

We construct an algorithm $\mathcal{R}$ which inverts RSA using a forger $\mathcal{F}$. The reduction $\mathcal{R}$ will answer by itself the hash queries and signature queries of $\mathcal{F}$. We assume that when the

forger makes a signature query he has already made the corresponding hash query. If not, the reduction goes ahead and makes the corresponding hash query. Similarly, we assume that the message M which signature is forged by the forger, has already been queried for hashing. Otherwise the reduction makes the corresponding hash query and proceeds.

Algorithm for $\mathcal{R}$:
Input: (N, e, y) and (q_{hash}, q_{sig}), where $(N, e) \leftarrow RSA(1^k)$ and $y \stackrel{R}{\leftarrow} \mathbb{Z}_N^*$.
Output: $y^d \bmod N$.

1. Set $i \leftarrow 0$
2. Send (N, e) to $\mathcal{F}$.
3. If $\mathcal{F}$ makes a hash query for M:
   $i \leftarrow i + 1$; $M_i \leftarrow M$; $r_i \stackrel{R}{\leftarrow} \mathbb{Z}_N^*$
   Flip a coin c_i with bias γ.
   $c_i = 0$ with probability γ and $c_i = 1$ with probability $1 - \gamma$.
   Return $H(M) = y^{c_i} \cdot r_i^e \bmod N$
4. If $\mathcal{F}$ makes a signature query for M_i:
   Return r_i if $H(M_i) = r_i^e \bmod N$. Otherwise stop.
5. If $\mathcal{F}$ outputs a forgery (M, x):
   If $H(M) = y \cdot r_i^e \bmod N$ then output $y^d = x/r_i \bmod N$. Otherwise stop.
6. Go to step 3

$\mathcal{R}$ answers a signature query at step 4 with probability γ; the probability that $\mathcal{R}$ answers all the signature queries is greater than $\gamma^{q_{sig}}$. Eventually $\mathcal{F}$ outputs a forgery with probability ε_F; $\mathcal{R}$ can use this forgery at step 5 with probability $1 - \gamma$ to output $y^d \bmod N$. Consequently $\mathcal{R}$ outputs $y^d \bmod N$ with probability $\gamma^{q_{sig}} \cdot (1 - \gamma) \cdot \varepsilon_F$, which is maximal for $\gamma = 1 - 1/(q_{sig} + 1)$ and gives (2).

B Proof of Theorem 3

B.1 A Variant of PSS

We describe a variant of PSS, which we call PFDH, for Probabilistic Full Domain Hash, for which the security proof is simpler. The scheme is similar to Full Domain Hash except that a random salt of k_0 bits is concatenated to the message M before hashing it. The difference with PSS is that the random salt is not recovered when verifying the signature; instead the random salt is transmitted separately. As FDH, the scheme uses a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_N^*$.

$\texttt{SignPFDH}(M)$:	$\texttt{VerifyPFDH}(M, s, r)$:
$r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$	$y \leftarrow s^e \bmod N$
$y \leftarrow H(M\|r)$	if $y = H(M\|r)$ then return 1
return $(y^d \bmod N, r)$	else return 0

The following theorem proves the security of PFDH in the random oracle model, assuming that inverting RSA is hard. It shows that PFDH has a tight security proof for a random salt of length $k_0 = \log_2 q_{sig}$ bits.

Theorem 8. *Suppose that RSA is (t', ε')-secure. Then the signature scheme* PFDH$[k_0]$ *is $(t, q_{hash}, q_{sig}, \varepsilon)$-secure, where:*

$$t = t' - (q_{hash} + q_{sig}) \cdot \mathcal{O}(k^3) - q_{hash} \cdot q_{sig} \cdot \mathcal{O}(k) \tag{16}$$

$$\varepsilon = \varepsilon' \cdot \left(1 + 6 \cdot q_{sig} \cdot 2^{-k_0}\right) \tag{17}$$

Proof. Let $\mathcal{F}$ be a forger which $(t, q_{sig}, q_{hash}, \varepsilon)$-breaks PFDH. We construct an inverter I which (t', ε')-breaks RSA. The inverter receives as input (N, e, η) and must output $\eta^d \bmod N$. We assume that the forger never repeats a hash query. However, the forger may repeat a signature query, in order to obtain the signature of M with distinct integers r. The inverter $\mathcal{I}$ maintains a counter i, initially set to zero.

When a message M appears for the first time in a hash query or a signature query, the inverter increments the counter i and sets $M_i \leftarrow M$. Then, the inverter generates a list L_i of q_{sig} random integers in $\{0,1\}^{k_0}$.

When the forger makes a hash query for $M_i \| r$, we distinguish two cases. If r belongs to the list L_i, the inverter generates a random $x \in \mathbb{Z}_N^*$ and returns $H(M_i \| r) = x^e \bmod N$. Otherwise, the inverter generates a random $x \in \mathbb{Z}_N^*$ and returns $\eta \cdot x^e \bmod N$. Consequently, for each message M_i, the list L_i contains the integers $r \in \{0,1\}^{k_0}$ such that the inverter knows the signature x corresponding to $M_i \| r$.

When the forger makes a signature query for M_i, the inverter takes the next random r in the list L_i. Since the list contains initially q_{sig} integers and there are at most q_{sig} signature queries, this is always possible. If there was already a hash query for $M_i \| r$, we have $H(M_i \| r) = x^e \bmod N$ and the inverter returns the signature x. Otherwise the inverter generates a random $x \in \mathbb{Z}_N^*$, sets $H(M_i \| r) = x^e \bmod N$ and returns the signature x.

When the forger outputs a forgery (M, s, r), we assume that it has already made a hash query for M, so $M = M_i$ for a given i. Otherwise, the inverter goes ahead and makes the hash query for $M \| r$. Then if r does not belong to the list L_i, we have $H(M_i \| r) = \eta \cdot x^e \bmod N$. From $s = H(M_i \| r)^d = \eta^d \cdot x \bmod N$, we obtain $\eta^d = s/x \bmod N$ and the inverter succeeds in outputting $\eta^d \bmod N$.

Since the forger has not made any signature query for the message M_i in the forgery (M_i, s, r), the forger has no information about the q_{sig} random integers in the list L_i. Therefore, the probability that r does not belong to L_i is $(1 - 2^{-k_0})^{q_{sig}}$. If the size k_0 of the random salt is greater than $\log_2 q_{sig}$, we obtain if $q_{sig} \geq 2$:

$$\left(1 - 2^{-k_0}\right)^{q_{sig}} \geq \left(1 - \frac{1}{q_{sig}}\right)^{q_{sig}} \geq \frac{1}{4}$$

Since the forger outputs a forgery with probability ε, the success probability ε' of the inverter is then at least $\varepsilon/4$, which shows that for $k_0 \geq \log_2 q_{sig}$ the probability of breaking PFDH is almost the same as the probability of inverting RSA.

For the general case, *i.e.* if we do not assume $k_0 \geq \log_2 q_{sig}$, we generate fewer than q_{sig} random integers in the list L_i, so that the salt r in the forgery (M_i, s, r) belongs to L_i with lower probability. More precisely, starting from an empty list L_i, the inverter generates with probability β a random $r \leftarrow \{0,1\}^{k_0}$, adds it to L_i, and starts again until the list

L_i contains q_{sig} elements. Otherwise (so with probability $1-\beta$) the inverter stops adding integers to the list. The number a_i of integers in L_i is then a random variable following a geometric law of parameter β:

$$\Pr[a_i = j] = \begin{cases} (1-\beta)\cdot\beta^j & \text{if } j < q_{sig} \\ \beta^{q_{sig}} & \text{if } j = q_{sig} \end{cases} \tag{18}$$

The inverter answers a signature query for M_i if the corresponding list L_i contains one more integer, which happens with probability β (otherwise the inverter must abort). Consequently, the inverter answers all the signature queries with probability greater than $\beta^{q_{sig}}$. Note that if $\beta = 1$, the setting boils down to the previous case: all the lists L_i contain exactly q_{sig} integers, and the inverter answers all the signature queries with probability one.

The probability that r in the forgery (M_i, s, r) does not belong to the list L_i is then $(1-2^{-k_0})^j$, when the length a_i of L_i is equal to j. The probability that r does not belong to L_i is then:

$$f(\beta) = \sum_{j=0}^{q_{sig}} \Pr[a_i = j] \cdot \left(1 - 2^{-k_0}\right)^j \tag{19}$$

Since the forger outputs a forgery with probability ε, the success probability of the inverter is at least $\varepsilon \cdot \beta^{q_{sig}} \cdot f(\beta)$. We select a value of β which maximizes this success probability; in section B.2 we show that for any (q_{sig}, k_0), there exists β_0 such that:

$$\beta_0^{q_{sig}} \cdot f(\beta_0) \geq \frac{1}{1 + 6 \cdot q_{sig} \cdot 2^{-k_0}} \tag{20}$$

which gives (17). The running time of $\mathcal{I}$ is the running time of $\mathcal{F}$ plus the time necessary to compute the integers $x^e \bmod N$ and to generate the lists L_i, which gives (16).

B.2 Proof of Inequality (20)

Let

$$g(\beta) = \beta^{q_{sig}} \cdot f(\beta) \tag{21}$$

We denote $g_0 = \max\{g(\beta); \beta \in [0,1]\}$ and want to prove that

$$g_0 \geq \frac{1}{1 + 6 \cdot q_{sig} \cdot 2^{-k_0}}$$

Denoting $\gamma = 2^{-k_0}$, we obtain from (18), (19) and (21):

$$g(\beta) = \frac{\beta^{q_{sig}}}{1 - (1-\gamma)\cdot\beta} \cdot \left(1 - \beta + \gamma \cdot (1-\gamma)^{q_{sig}} \cdot \beta^{q_{sig}+1}\right) \tag{22}$$

from which we derive:

$$g(\beta) \geq \beta^{q_{sig}} \cdot \frac{1-\beta}{1-\beta+\gamma}$$

If $\gamma \cdot q_{sig} \geq 1/2$, we take $\beta = 1 - 1/(2 \cdot q_{sig})$ and obtain:

$$g_0 \geq \left(1 - \frac{1}{2 \cdot q_{sig}}\right)^{q_{sig}} \cdot \frac{1}{1 + 2 \cdot \gamma \cdot q_{sig}}$$

For $q_{sig} \geq 1$ we have

$$\left(1 - \frac{1}{2 \cdot q_{sig}}\right)^{q_{sig}} \geq \frac{1}{2}$$

Using $\gamma \cdot q_{sig} \geq 1/2$, we obtain

$$g_0 \geq \frac{1}{2 \cdot (1 + 2 \cdot \gamma \cdot q_{sig})} \geq \frac{1}{1 + 6 \cdot \gamma \cdot q_{sig}}$$

For $\gamma \cdot q_{sig} \leq 1/2$, we take $\beta = 1$ and obtain using (22):

$$g_0 \geq (1 - \gamma)^{q_{sig}} \geq 1 - \gamma \cdot q_{sig} \geq \frac{1}{1 + 6 \cdot \gamma \cdot q_{sig}} \quad \text{for } \gamma \cdot q_{sig} \leq 1/2$$

B.3 Proof of Theorem 3

Let $\mathcal{F}$ be a forger which $(t, q_{sig}, q_{hash}, \varepsilon)$-breaks PSS. We construct an inverter I which (t', ε')-breaks RSA. The inverter receives as input (N, e, η) and must output $\eta^d \bmod N$. The inverter $\mathcal{I}$ maintains a counter i, initially 0.

The proof is very similar to the proof of theorem 8 and to the initial security proof of PSS in [2]. To answer a hash query $M||r$ in theorem 8, we generated a random $x \in \mathbb{Z}_N$ and $y = x^e \cdot \eta^b$ with $b = 0$ or $b = 1$, and defined $H(M||r) = y$. The only difference here is that we write y as $0||\omega||r^*||\gamma$, where the size of ω is k_1 bits, the size of r^* is k_0 bits and the size of γ is the remaining $k - k_0 - k_1 - 1$ bits. We define $H(M||r) = \omega$ and $G(\omega) = r^* \oplus r||\gamma$. Moreover we must make sure that the same ω never appears twice otherwise we would be re-defining $G(\omega)$.

When a message M appears for the first time in a hash query or a signature query, the inverter increments the counter i and sets $M_i \leftarrow M$. Then, the inverter generates a list L_i of q_{sig} random integers in $\{0,1\}^{k_0}$.

When the forger makes a H-oracle query for $M_i\|r$, we distinguish two cases. If r belongs to the list L_i, the inverter sets $b = 0$, else it sets $b = 1$. Then the inverter generates a random $x \in \mathbb{Z}_N^*$ until the first bit of $y = x^e \cdot \eta^b \bmod N$ is 0. Then it writes y as $0\|\omega\|r^*\|\gamma$ and sets $H(M_i\|r) = \omega$. The inverter aborts if ω has already appeared before. Eventually the inverter sets $G(\omega) = r^* \oplus r\|\gamma$ and returns ω as the answer to the H-oracle query $M_i\|r$.

When the forger makes a G-oracle query for ω, the inverter returns $G(\omega)$ if ω appeared before. Otherwise it generates a random string $\alpha \leftarrow \{0,1\}^{k-k_1-1}$, sets $G(\omega) = \alpha$, and returns α.

When the forger makes a signature query for M_i, the inverter takes the next random r in L_i. If there was already a H-oracle query for $M_i\|r$, the inverter knows x, y, ω, r^* and γ such that $y = x^e \bmod N$ and $y = 0\|\omega\|r^*\|\gamma$ where $H(M_i||r) = \omega$ and $G(\omega) = r^* \oplus r\|\gamma$, so the inverter returns x as a signature for M_i. Otherwise, the inverter generates a random $x \in \mathbb{Z}_N^*$ until the first bit of $y = x^e \bmod N$ is 0. Then it writes y as $0\|\omega\|r^*\|\gamma$ and sets $H(M_i\|r) = \omega$. The inverter aborts if ω has already appeared before. Then the inverter sets $G(\omega) = r^* \oplus r\|\gamma$, and returns x as a signature for M_i.

Since there are at most q_{hash} hash queries and q_{sig} signature queries, the number of distinct ω which can appear is less than $q_{hash} + q_{sig}$. The probability that the inverter aborts after generating a random ω is then less than $(q_{hash} + q_{sig}) \cdot 2^{-k_1}$. Therefore, the inverter aborts when answering the hash and signature queries with probability less than $\delta = (q_{hash} + q_{sig})^2 \cdot 2^{-k_1}$. Consequently, the forger outputs a forgery with probability at least $\varepsilon - \delta$.

When the forger outputs a forgery (M, s), we compute $y = s^e \bmod N$ and write y as $0\|\omega\|r^*\|\gamma$. Let $r = r^* \oplus G_1(\omega)$, where G_1 denotes the first k_0 bits of G. If there was no H-oracle query for $M\|r$ before, the probability that $\omega = H(M\|r)$ is at most 2^{-k_1}. Therefore, with probability at least $\varepsilon - \delta - 2^{-k_1}$, the forger outputs a forgery and there exists an integer i such that there has been a H-oracle query for $M_i\|r$. Then if r does not belong to the list L_i, the inverter knows x such that $y = x^e \cdot \eta$, which gives $\eta^d = s/x \bmod N$ and the inverter succeeds in outputting $\eta^d \bmod N$.

As in theorem 8, the probability that r does not belong to the list L_i of q_{sig} random integers is $(1 - 2^{-k_0})^{q_{sig}}$. If $k_0 \geq \log_2 q_{sig}$ and for $q_{sig} \geq 2$, this gives

$$\left(1 - 2^{-k_0}\right)^{q_{sig}} \geq \left(1 - \frac{1}{q_{sig}}\right)^{q_{sig}} \geq \frac{1}{4}$$

Consequently, the success probability ε' of the inverter is at least $(\varepsilon - \delta - 2^{-k_1})/4$, which shows that for $k_0 \geq \log_2 q_{sig}$ the probability of breaking $\mathtt{PSS}[k_0, k_1]$ is almost the same as the probability of inverting RSA.

For smaller values of k_0, we apply the same trick as in theorem 8: we generate fewer than q_{sig} random integer in the lists L_i, according to the same distribution with parameter β. As in theorem 8, the success probability of the inverter is at least:

$$\left(\varepsilon - \delta - 2^{-k_1}\right) \cdot \beta^{q_{sig}} \cdot f(\beta)$$

where $f(\beta)$ is given by equation (19). As in theorem 8, we select a value of β which maximizes this success probability; we obtain that the inverter succeeds with probability at least:

$$\frac{\varepsilon - \delta - 2^{-k_1}}{1 + 6 \cdot q_{sig} \cdot 2^{-k_0}}$$

Moreover, when answering the hash and signature queries, the probability that the first bit of $x^e \cdot \eta^b \bmod N$ is 0 for a random $x \in \mathbb{Z}_N$ is at least $1/2$. Therefore we stop the loop after $1 + k_1$ steps[3], which adds a failure probability of 2^{-k_1} per hash or signature query. Eventually, the success probability ε' of the inverter is at least:

$$\varepsilon' = \frac{\varepsilon - 2 \cdot (q_{hash} + q_{sig})^2 \cdot 2^{-k_1}}{1 + 6 \cdot q_{sig} \cdot 2^{-k_0}}$$

which gives equation (7).

The space required to store the lists L_i is $q_{hash} \cdot q_{sig} \cdot \mathcal{O}(k)$, whereas the inverter of [2] required a space in $(q_{hash} + q_{sig}) \cdot \mathcal{O}(k)$ only. To obtain the same space complexity,

[3] otherwise the running time could not be bounded.

one could use a pseudo-random number generator: instead of storing the entire list L_i, we store a seed s_i for each message M_i. When we receive a hash query for $M_i||r$, using s_i we generate the list L_i to determine if r belongs to L_i. However the time complexity would still be proportional to $q_{hash} \cdot q_{sig}$, whereas the time complexity in [2] is linear in $q_{hash} + q_{sig}$. To obtain the same time complexity as in [2], we build the lists L_i in the process of answering the hash and signature queries for M_i, instead of generating the lists at the beginning. We briefly sketch how this can be done.

The list L_i is initially empty. Let b_i be its current length, and let a_i be its final length, where a_i is a random variable following (18). Let U_i be a list, initially empty, of integers which are not in L_i, and let u_i be the size of U_i, so initially $b_i = 0$ and $u_i = 0$. When the forger makes a H-oracle query for $M_i||r$, let α be the probability that r belongs to the final list L_i. Then with probability α, we include r in the current list L_i and let $b_i \leftarrow b_i + 1$. We store r together with its randomly generated index $j \in [1; a_i]$ in L_i. The index j in L_i must not have been selected before. Otherwise (with probability $1 - \alpha$), we include r in the list U_i of integers not in L_i, and let $u_i \leftarrow u_i + 1$. In both cases the hash query is then answered as previously.

When the forger makes the j-th signature query for M_i, we abort if $j > a_i$. Otherwise we select the integer r with index j in L_i, if such integer exists. Otherwise, we generate a random r in $\{0,1\}^{k_0} \setminus U_i$. The signature query is then answered as previously.

It remains to compute α. We assume that the forger never repeats a hash query, so when the forger makes a H-oracle query for $M_i||r$, the integer r does not belong to the current list L_i of length b_i. The probability that r belongs to the final list L_i of length a_i is then the probability that an integer in $\{0,1\}^{k_0}$ belongs to a list of $a_i - b_i$ random integers in $\{0,1\}^{k_0} \setminus U_i$, so :

$$\alpha = 1 - \left(1 - \frac{1}{2^{k_0} - u_i}\right)^{a_i - b_i}$$

The running time of the inverter is then the running time of $\mathcal{F}$ plus the time necessary to compute the integers $x^e \bmod N$ and to generate the lists L_i, which gives (16).

C Proof of Theorem 4

From $\mathcal{R}$ we build an algorithm $\mathcal{I}$ which inverts RSA, without using a forger for FDH. We receive as input (N, e, y) and our goal is to output $y^d \bmod N$ using $\mathcal{R}$. We select q_{hash} distinct messages $M_1, \ldots, M_{q_{hash}}$ and start running $\mathcal{R}$ with (N, e, y).

First we ask $\mathcal{R}$ to hash the q_{hash} messages $M_1, \ldots, M_{q_{hash}}$, and obtain the hash values $h_1, \ldots, h_{q_{hash}}$. We select a random integer $\beta \in [1, q_{hash}]$ and a random sequence α of q_{sig} integers in $[1, q_{hash}] \setminus \{\beta\}$, which we denote $\alpha = (\alpha_1, \ldots, \alpha_{q_{sig}})$. We select a random integer $i \in [1, q_{sig}]$ and define the sequence of i integers $\alpha' = (\alpha_1, \ldots, \alpha_{i-1}, \beta)$. Then we make the i signature queries corresponding to α' to $\mathcal{R}$ and receive from $\mathcal{R}$ the corresponding signatures, the last one being the signature s_β of M_β. For example, if $\alpha' = (3, 2)$, this corresponds to making a signature query for M_3 first, and then for M_2.

Then we rewind $\mathcal{R}$ to the state it was after the hash queries, and this time, we make the q_{sig} signature queries corresponding to α. If $\mathcal{R}$ has answered all the signature queries, then with probability ε_F, we send (M_β, s_β) as a forgery to $\mathcal{R}$. This is a true forgery for $\mathcal{R}$

because after the rewind of $\mathcal{R}$, there was no signature query for M_β. Eventually $\mathcal{R}$ inverts RSA and outputs $y^d \bmod N$.

We denote by $\mathcal{Q}$ the set of sequences of signature queries which are correctly answered by $\mathcal{R}$ after the hash queries, in time less than t_R. If a sequence of signature queries is correctly answered by $\mathcal{R}$, then the same sequence without the last signature query is also correctly answered, so for any $(\alpha_1, \ldots, \alpha_j) \in \mathcal{Q}$, we have $(\alpha_1, \ldots, \alpha_{j-1}) \in \mathcal{Q}$. Let us denote by `ans` the event $\alpha \in \mathcal{Q}$, which corresponds to $\mathcal{R}$ answering all the signature queries after the rewind, and by `ans'` the event $\alpha' \in \mathcal{Q}$, which corresponds to $\mathcal{R}$ answering all the signature queries before the rewind.

Let us consider a forger which makes the same hash queries, the same signature queries corresponding to α, and outputs a forgery for M_β with probability ε_F. By definition, when interacting with such a forger, $\mathcal{R}$ would output $y^d \bmod N$ with probability at least ε_R. After the rewind, $\mathcal{R}$ sees exactly the same transcript as when interacting with this forger, except if event `ans` is true and `ans'` is false: in this case, the forger outputs a forgery with probability ε_F, whereas our simulation does not output a forgery. Consequently, when interacting with our simulation of a forger, $\mathcal{R}$ outputs $y^d \bmod N$ with probability at least:

$$\varepsilon_R - \varepsilon_F \cdot \Pr[\texttt{ans} \wedge \neg \texttt{ans'}] \tag{23}$$

Lemma 1. *Let $\mathcal{Q}$ be a set of sequences of at most n integers in $[1, k]$, such that for any sequence $(\alpha_1, \ldots, \alpha_j) \in \mathcal{Q}$, we have $(\alpha_1, \ldots, \alpha_{j-1}) \in \mathcal{Q}$. Then the following holds:*

$$\Pr_{\substack{i \leftarrow [1,n] \\ (\alpha_1, \ldots, \alpha_n, \beta) \leftarrow [1,k]^{n+1}}} [(\alpha_1, \ldots, \alpha_n) \in \mathcal{Q} \wedge (\alpha_1, \ldots, \alpha_{i-1}, \beta) \notin \mathcal{Q}] \leq \frac{\exp(-1)}{n}$$

Proof. The proof is given in appendix D.

Using lemma 1 with $n = q_{sig}$ and $k = q_{hash}$, we obtain:

$$\Pr[\texttt{ans} \wedge \neg \texttt{ans'}] \leq \frac{\exp(-1)}{q_{sig}} \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1} \tag{24}$$

The term $(1 - q_{sig}/q_{hash})$ in equation (24) is due to the fact that we select $\alpha_1, \ldots, \alpha_{q_{sig}}$ in $[1, q_{hash}] \setminus \{\beta\}$ whereas in lemma 1 the integers are selected in $[1, q_{hash}]$. From equations (23) and (24) we obtain that $\mathcal{I}$ succeeds with probability greater than ε_I given by (9). Because of the rewind, the running time of $\mathcal{I}$ is at most twice the running time of $\mathcal{R}$, which gives (8).

D Proof of Lemma 1

We show inductively over n that, letting D_n be the following distribution

$$D_n = \left\{ \begin{array}{r} i \leftarrow [1, n] \\ (\alpha_1, \ldots, \alpha_n) \leftarrow [1, k]^n \\ \beta \leftarrow [1, k] \end{array} \right.$$

and denoting for any $j \in [1, n]$ the events:

$$A_j : (\alpha_1, \ldots, \alpha_{j-1}, \alpha_j) \in Q$$
$$B_j : (\alpha_1, \ldots, \alpha_{j-1}, \beta) \in Q$$

with $A_j \Rightarrow A_{j-1}$ for all $j \in [2, n]$, then the following holds:

$$\Pr_{D_n}[A_n \wedge B_i] \geq \Pr_{D_n}[A_n]^{1+\frac{1}{n}} \tag{25}$$

Inequality (25) clearly holds for $n = 1$. Assuming that inequality (25) holds for $n-1$, we show that it holds for n. In the following, unless specified otherwise, probabilities are taken according to the distribution D_n. Since i is randomly selected in $[1, n]$, we have:

$$\Pr[A_n \wedge B_i] = \frac{1}{n}\Pr[A_n \wedge B_1] + \frac{n-1}{n}\Pr[A_n \wedge B_i | i \geq 2] \tag{26}$$

The events A_n and B_1 are independent, which gives:

$$\Pr[A_n \wedge B_1] = \Pr[A_n] \cdot \Pr[B_1] = \Pr[A_n] \cdot \Pr[A_1] \tag{27}$$

We have:

$$\Pr[A_n] = \frac{1}{k} \sum_{a_1 \in [1,k]} \Pr[A_n | \alpha_1 = a_1]$$

and

$$\Pr[A_n \wedge B_i | i \geq 2] = \frac{1}{k} \sum_{a_1 \in [1,k]} \Pr\left[A_n \wedge B_i | (\alpha_1 = a_1) \wedge (i \geq 2)\right]$$

Letting $L_1 = \{a_1 \in [1, k] \mid (a_1) \in Q\}$, we have using $\Pr[A_1] = \#L_1/k$ and $A_n \Rightarrow A_1$:

$$\Pr[A_n | A_1] = \frac{\Pr[A_n \wedge A_1]}{\Pr[A_1]} = \frac{\Pr[A_n]}{\Pr[A_1]} = \frac{1}{\#L_1} \sum_{a_1 \in L_1} \Pr[A_n | \alpha_1 = a_1] \tag{28}$$

and

$$\Pr[A_n \wedge B_i | i \geq 2] = \Pr[A_1] \cdot \frac{1}{\#L_1} \sum_{a_1 \in L_1} \Pr\left[A_n \wedge B_i | (\alpha_1 = a_1) \wedge (i \geq 2)\right] \tag{29}$$

For all $j \in [2, n]$, let $A'_{j-1} = A_j \wedge (\alpha_1 = a_1)$ and $B'_{j-1} = B_j \wedge (\alpha_1 = a_1)$, and let D'_{n-1} be the following distribution:

$$D'_{n-1} = \left\{ \begin{array}{r} i' \leftarrow [1, n-1] \\ (\alpha_2, \ldots, \alpha_n) \leftarrow [1, k]^{n-1} \\ \beta \leftarrow [1, k] \end{array} \right.$$

We have:

$$\Pr_{D'_{n-1}}[A'_{n-1}] = \Pr[A_n | \alpha_1 = a_1] \tag{30}$$

and

$$\Pr_{D'_{n-1}}[A'_{n-1} \wedge B'_{i'}] = \Pr\left[A_n \wedge B_i | (\alpha_1 = a_1) \wedge (i \geq 2)\right] \tag{31}$$

Applying inequality (25) for $n-1$, we obtain:

$$\Pr_{D'_{n-1}}[A'_{n-1} \wedge B'_{i'}] \geq \Pr_{D'_{n-1}}[A'_{n-1}]^{\frac{n}{n-1}}$$

which gives using equations (29), (30) and (31):

$$\Pr[A_n \wedge B_i | i \geq 2] \geq \Pr[A_1] \cdot \frac{1}{\#L_1} \sum_{a_1 \in L_1} \Pr[A_n | \alpha_1 = a_1]^{\frac{n}{n-1}} \tag{32}$$

From the inequality

$$\frac{1}{t}\sum_{i=1}^{t} x_i^r \geq \left(\frac{1}{t}\sum_{i=1}^{t} x_i\right)^r \quad \text{for} \quad r \geq 1$$

we obtain:

$$\Pr[A_n \wedge B_i | i \geq 2] \geq \Pr[A_1] \cdot \left(\frac{1}{\#L_1} \sum_{a_1 \in L_1} \Pr[A_n | \alpha_1 = a_1]\right)^{\frac{n}{n-1}}$$

which gives using (28):

$$\Pr[A_n \wedge B_i | i \geq 2] \geq \Pr[A_1] \cdot \Pr[A_n | A_1]^{\frac{n}{n-1}} = \Pr[A_n] \cdot \Pr[A_n | A_1]^{\frac{1}{n-1}}$$

Then using equations (26) and (27), we obtain:

$$\Pr[A_n \wedge B_i] \geq \Pr[A_n] \left(\frac{\Pr[A_1]}{n} + \frac{n-1}{n} \Pr[A_n | A_1]^{\frac{1}{n-1}}\right)$$

Using the well known inequality $S \geq P$ between the arithmetic mean S and the geometric mean P, we obtain:

$$\frac{1}{n}\left(\Pr[A_1] + (n-1) \cdot \Pr[A_n | A_1]^{\frac{1}{n-1}}\right) \geq (\Pr[A_1] \cdot \Pr[A_n | A_1])^{\frac{1}{n}} = \Pr[A_n]^{\frac{1}{n}}$$

and eventually

$$\Pr[A_n \wedge B_i] \geq \Pr[A_n]^{1+\frac{1}{n}}$$

which shows that equation (25) holds for n and terminates the proof by induction.

Inequality (25) gives:

$$\Pr[A_n \wedge \neg B_i] = \Pr[A_n] - \Pr[A_n \wedge B_i] \leq \Pr[A_n] \cdot \left(1 - \Pr[A_n]^{1/n}\right)$$

Denoting $x = \Pr[A_n]^{1/n}$ and using the inequality $x^n \cdot (1-x) \leq \exp(-1)/n$ for $x \in [0,1]$, we obtain:

$$\Pr[A_n \wedge \neg B_i] \leq \frac{\exp(-1)}{n}$$

E Definitions for Security Proofs in the Random Oracle Model

In this section, we consider a signature scheme provably secure in the random oracle model. In the random oracle model, the hash function is replaced by a random function.

Definition 8 (random oracle). *For any constant k, a random oracle is a function H selected uniformly at random in the set $\mathcal{H}_k$ of the functions from $\{0,1\}^*$ to $\{0,1\}^k$.*

We say that a signature scheme is a hash-and-sign signature scheme if the signature generation algorithm first hashes the message and then signs the hash value using the private key.

Definition 9 (hash-and-sign scheme). *A signature scheme* (`Gen`, `Sign`, `Verify`) *is said to be a hash-and-sign signature scheme if* `Sign` *takes as input the message M, the public key pk and the private key sk, runs* `Hashing` *with M and pk, obtains h, then runs* `Signing` *with h and sk, obtains and returns the signature x, where:*
- `Hashing` is an algorithm taking as input the message M to be signed and the public key pk and returning a string h. `Hashing` *may have access to a random oracle.*
- `Signing` is an algorithm taking as input h and the private key sk and returning the signature x. `Signing` *does not have access to a random oracle.*

Examples of hash-and-sign signature schemes include the FDH scheme, PSS, Gennaro-Halevi-Rabin's scheme (GHR) in the random oracle model [11], Paillier's signature scheme [16] and DSA [10].

The `hashing` algorithm may require multiple hash oracle queries, for example two hash queries as in PSS. For simplicity, we say in the following that a forger can make q_{hash} hash queries if he can apply `Hashing` to q_{hash} messages. The actual number of hash queries q'_{hash} will then be a multiple of q_{hash} (for PSS, we have $q'_{hash} = 2 \cdot q_{hash}$).

We say that a signature scheme is with unique signature if each message has a unique signature, given the random oracle $H \in \mathcal{H}_k$; formally:

Definition 10 (signature scheme with unique signature). *A signature scheme is said to be with unique signature if for any public key pk, any message M and any random oracle H in $\mathcal{H}_k$, there is a unique x such that* $\texttt{Verify}_{pk}(M,x) = 1$.

Hash-and-sign signature schemes with unique signature include FDH, GHR in the random oracle model and Paillier's signature scheme. PSS and DSA are *not* signature schemes with unique signature.

Lemma 2. *Let $\mathcal{S}$ be a hash-and-sign signature scheme with unique signature. Let $h \leftarrow \texttt{Hashing}_{pk}(M)$. The signature x of M is then a function of h and the public key pk only.*

Proof. We denote $\texttt{Sign}^H_{pk,sk}$, $\texttt{Hashing}^H_{pk}$ and $\texttt{Verify}^H_{pk}$ the algorithms `Sign`, `Hashing` and `Verify` with oracle access to $H \in \mathcal{H}_k$.

Let x be the signature of M with random oracle $H \in \mathcal{H}_k$ and public key pk. Let (pk', sk') be another public/private key pair, M' another message, and $H' \in \mathcal{H}_k$ another random oracle. Let $h' \leftarrow \texttt{Hashing}^{H'}_{pk'}(M')$ and $x' \leftarrow \texttt{Signing}_{pk',sk'}(h')$. We must show that if $pk = pk'$ and $h = h'$, then $x = x'$.

From $pk = pk'$ and $h = h'$, we deduce $h' \leftarrow \mathtt{Hashing}^H_{pk}(M)$ and $x' \leftarrow \mathtt{Signing}_{pk,sk'}(h')$, which implies that x' is a signature of M under the public key pk with random oracle H. Since $\mathcal{S}$ is a signature scheme with unique signature, we must have $x = x'$.

□

The security of the signature scheme that we consider is not necessarily based on the hardness of inverting RSA; it can be based on the hardness of any search problem Π, defined as follows:

Definition 11. *A search problem Π is a triple $(\mathtt{Gen}\Pi, D, S)$ where D is a set of finite objects called instances, and for each instance $I \in D$, $S[I]$ is a set of finite objects called solutions for I. $\mathtt{Gen}\Pi$ is an algorithm which, on input 1^k, randomly selects an instance $I \in D$ such that $|I| = k$.*

Definition 12. *An algorithm $\mathcal{A}$ is said to (t_A, ε_A)-solve Π if after receiving an instance I generated using $\mathtt{Gen}\Pi(1^k)$ and $t_A(k)$ processing time it outputs a solution z for I with probability greater than $\varepsilon_A(k)$ for all $k \in \mathbb{N}$.*

Definition 13. *A problem Π is said to be (t_A, ε_A)-hard if there is no algorithm $\mathcal{A}$ which (t_A, ε_A)-solves Π.*

In the following we consider a hash-and-sign signature scheme with unique signature provably secure in the random oracle model, assuming that solving a given problem Π is hard. This means that there exists a reduction from solving the hard problem Π to breaking the signature scheme $\mathcal{S}$. A reduction from solving Π to breaking $\mathcal{S}$ is an algorithm which uses a forger for $\mathcal{S}$ in order to solve Π. Resistance against adaptive chosen message attacks is considered, so the forger is allowed to make signature queries for messages of his choice. Moreover, in the random oracle model, the forger cannot compute the hash function by itself: the forger must make a hash query. Consequently, when interacting with the forger, the reduction algorithm must answer the hash queries and the signature queries made by the forger.

Definition 14. *A reduction R in the random oracle model from solving $(Gen\Pi, D, S)$ to breaking (*Gen, Sign, Verify*) is a probabilistic algorithm taking as input an instance I and (q_{hash}, q_{sig}), where $I \leftarrow Gen\Pi(1^k)$, and outputting a solution z for I. The reduction algorithm interacts with a forger $\mathcal{F}$ for (*Gen, Sign, Verify*) which outputs a forgery after at most q_{hash} hash queries and q_{sig} signature queries. The reduction algorithm answers the hash queries and the signature queries made by $\mathcal{F}$.*

We quantify the efficiency of the reduction by giving the probability that the reduction algorithm outputs a solution of the problem Π using a forger that $(t_F, q_{hash}, q_{sig}, \varepsilon_F)$-breaks the signature scheme, within an additional running time of t_R.

Definition 15. *We say that a reduction algorithm $\mathcal{R}$ $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces solving $(\mathtt{Gen}\Pi, D, S)$ to breaking the signature scheme (*Gen, Sign, Verify*) if after running any forger that $(t_F, q_{hash}, q_{sig}, \varepsilon_F)$-breaks (*Gen, Sign, Verify*), the reduction outputs a solution of Π with probability greater than ε_R, within an additional running time of t_R.*

In this section we consider reductions running a forger only once, as the reduction of theorem 1 for FDH. Reductions running a forger more than once will be considered in the next section. The following theorem shows that for any hash-and-sign signature scheme with unique signature provably secure in the random oracle model, assuming the hardness of a given problem Π, the security reduction cannot be tighter than roughly ε_F/q_{sig}. Namely we show that from $\mathcal{R}$ we can solve the problem Π with probability greater than roughly $\varepsilon_R - \varepsilon_F/q_{sig}$, in roughly the same time bound. Thus, if solving Π is hard, the success probability ε_R of $\mathcal{R}$ cannot be greater than roughly ε_F/q_{sig}.

Theorem 9. *Let $\mathcal{R}$ be a reduction which $(t_R, q_{hash}, q_{sig}, \varepsilon_R, \varepsilon_F)$-reduces solving Π to breaking a hash-and-sign signature scheme with unique signature. $\mathcal{R}$ runs the forger only once. From $\mathcal{R}$ we can construct an algorithm which (t_A, ε_A)-solves Π, with:*

$$t_A = 2 \cdot t_R \tag{33}$$

$$\varepsilon_A = \varepsilon_R - \varepsilon_F \cdot \frac{\exp(-1)}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1} \tag{34}$$

Proof. From a reduction $\mathcal{R}$ that $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces solving Π to breaking the signature scheme $(\texttt{Gen}, \texttt{Sign}, \texttt{Verify})$, we build an algorithm $\mathcal{A}$ that (t_A, ε_A)-solves Π using $\mathcal{R}$. The algorithm $\mathcal{A}$ receives as input an instance I of the problem Π and must output a solution z of I using $\mathcal{R}$. As in the proof of theorem 4, the algorithm $\mathcal{A}$ will simulate a forger with respect to $\mathcal{R}$. $\mathcal{A}$ arbitrarily selects q_{hash} distinct messages $M_1, \ldots, M_{q_{hash}}$.

First $\mathcal{A}$ receives from $\mathcal{R}$ the public key pk. Then $\mathcal{A}$ runs `Hashing` for the q_{hash} messages $M_1, \ldots, M_{q_{hash}}$, performs the corresponding hash queries to $\mathcal{R}$, and obtains the corresponding strings $h_1, \ldots, h_{q_{hash}}$. $\mathcal{A}$ selects a random integer $\beta \in [1, q_{hash}]$ and a random sequence α of q_{sig} integers in $[1, q_{hash}] \setminus \{\beta\}$, which we denote $\alpha = (\alpha_1, \ldots, \alpha_{q_{sig}})$. $\mathcal{A}$ selects a random integer $i \in [1, q_{sig}]$ and define the sequence of i integers $\alpha' = (\alpha_1, \ldots, \alpha_{i-1}, \beta)$. Then $\mathcal{A}$ makes the i signature queries corresponding to α' to $\mathcal{R}$ and receive from $\mathcal{R}$ the corresponding signatures, the last one being the signature s_β of M_β.

Then the reduction $\mathcal{R}$ is rewound to the state in which it was after the hash queries, and this time, $\mathcal{A}$ makes the q_{sig} signature queries corresponding to α. If $\mathcal{R}$ has answered all the signature queries, then with probability ε_F, $\mathcal{A}$ sends (M_β, s_β) as a forgery to $\mathcal{R}$. From lemma 2 the signature s_β of M_β is a function of h_β and pk only, so s_β is still a valid signature of M_β after $\mathcal{R}$ has been rewound. This is a true forgery for $\mathcal{R}$ because after the rewind of $\mathcal{R}$, there was no signature query for M_β. Eventually $\mathcal{R}$ outputs a solution z of instance I.

We denote by $\mathcal{Q}$ the set of sequences of signature queries which are correctly answered by $\mathcal{R}$ after the hash queries, in time less than t_R. If a sequence of signature queries is correctly answered by $\mathcal{R}$, then the same sequence without the last signature query is also correctly answered, so for any $(\alpha_1, \ldots, \alpha_j) \in \mathcal{Q}$, we have $(\alpha_1, \ldots, \alpha_{j-1}) \in \mathcal{Q}$. Let denote by `ans` the event $\alpha \in \mathcal{Q}$, which corresponds to $\mathcal{R}$ answering all the signature queries after the rewind, and by `ans'` the event $\alpha' \in \mathcal{Q}$, which corresponds to $\mathcal{R}$ answering all the signature queries before the rewind.

Let consider a forger which makes the same hash queries, the same signature queries corresponding to α, and outputs a forgery for M_β with probability ε_F. By definition, when interacting with such a forger, $\mathcal{R}$ would output $y^d \bmod N$ with probability at least ε_R.

After the rewind, $\mathcal{R}$ sees exactly the same transcript as when interacting with this forger, except if event **ans** is true and **ans'** is false: in this case, the forger outputs a forgery with probability ε_F, whereas our simulation does not output a forgery. Consequently, when interacting with our simulation of a forger, $\mathcal{R}$ outputs $y^d \bmod N$ with probability at least:

$$\varepsilon_R - \varepsilon_F \cdot \Pr[\mathbf{ans} \wedge \neg \mathbf{ans'}] \tag{35}$$

Using lemma 1 with $n = q_{sig}$ and $k = q_{hash}$, we obtain:

$$\Pr[\mathbf{ans} \wedge \neg \mathbf{ans'}] \leq \frac{\exp(-1)}{q_{sig}} \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1} \tag{36}$$

The term $(1 - q_{sig}/q_{hash})$ in equation (36) is due to the fact that we select $\alpha_1, \ldots, \alpha_{q_{sig}}$ in $[1, q_{hash}] \setminus \{\beta\}$ whereas in lemma 1 the integers are selected in $[1, q_{hash}]$. From equations (35) and (36) we obtain that $\mathcal{I}$ succeeds with probability greater than ε_I given by (34). Because of the rewind, the running time of $\mathcal{I}$ is at most twice the running time of $\mathcal{R}$, which gives (33).

F Proof of Theorem 5

In this section, we consider reductions running a forger more than once, as opposed to section E in which the forger was run only once. The reduction can run or rewind the forger at most r times. Using the same technique as previously, we show that from a reduction $\mathcal{R}$ allowed to run or rewind the forger at most r times, we can solve the problem Π with probability greater than roughly $\varepsilon_R - \varepsilon_F \cdot r/q_{sig}$ in roughly the same time bound. Thus, if solving Π is hard, the success probability of $\mathcal{R}$ cannot be greater than roughly $\varepsilon_F \cdot r/q_{sig}$.

The proof is very similar to the proof of theorem 9. Assume first that the reduction is not allowed to rewind the forger. The reduction is only allowed to run the forger at most r times. We say that the reduction is in the j-th round if the reduction has already run the forger $j-1$ times. Thus there are at most r rounds.

In the first round of the reduction, we apply exactly the same technique as previously: we make the q_{hash} hash queries, then we select a random $\beta_1 \in [1, q_{hash}]$ and a random sequence α_1 of q_{sig} integers in $[1, q_{hash}] \setminus \{\beta_1\}$. We select a random integer $i_1 \in [1, q_{sig}]$ and define the sequence α'_1 as the first $i_1 - 1$ integers of α_1 plus the integer β_1. Then we make the i_1 signature queries corresponding to α'_1 to $\mathcal{R}$ and obtain the signature s_{β_1} of M_{β_1}. Then we rewind $\mathcal{R}$ to the state it was after the hash queries, and this time, we make the q_{sig} signature queries corresponding to α_1. If $\mathcal{R}$ has answered all the signature queries, then with probability ε_F, we send $(M_{\beta_1}, s_{\beta_1})$ as a forgery to $\mathcal{R}$.

Then the reduction is in the second round, and starts interacting with a forger for the second time. We proceed recursively for the remaining rounds: at the j-th round, we make the same q_{hash} hash queries and select β_j, α_j and i_j as previously. We obtain the signature of M_{β_j}, then we rewind $\mathcal{R}$ to the state it was after the hash queries, then make the signature queries corresponding to α_j, and with probability ε_R output the signature of M_{β_j} as a forgery. Using this technique, we are able to simulate a forger being run at most r times by the reduction.

Let us denote $\mathbf{ans}'_j$ the event in which the reduction in the j-th round answers the signature queries before it is rewound and $\mathbf{ans}_j$ the event in which the reduction answers the signature queries after it is has been rewound.

Let consider a forger which at each round makes the same hash queries and signature queries corresponding to α_j, and outputs a forgery for M_{β_j} with probability ε_F. By definition, when running at most r times this forger, $\mathcal{R}$ succeeds with probability at least ε_R.

After the rewind of the j-th round, $\mathcal{R}$ sees exactly the same transcript as when interacting with this forger, except if event $\mathbf{ans}_j$ is true and $\mathbf{ans}'_j$ is false: in this case, this forger outputs a forgery with probability ε_F, whereas our simulation does not output a forgery. This happens with probability:

$$\varepsilon_F \cdot \Pr[\mathbf{ans}_j \wedge \neg \mathbf{ans}'_j]$$

Since there are at most r rounds, $\mathcal{A}$ succeeds with probability greater than:

$$\varepsilon_R - \sum_{j=1}^{r} \varepsilon_F \cdot \Pr[\mathbf{ans}_j \wedge \neg \mathbf{ans}'_j]$$

Using lemma 1, we obtain for all j:

$$\Pr[\mathbf{ans}_j \wedge \neg \mathbf{ans}'_j] \leq \frac{\exp(-1)}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1}$$

Consequently, $\mathcal{A}$ succeeds with probability greater than:

$$\varepsilon_A = \varepsilon_R - \varepsilon_F \cdot \frac{\exp(-1) \cdot r}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{q_{hash}}\right)^{-1} \tag{37}$$

The reduction $\mathcal{R}$ is rewound at most r times, so the running time of $\mathcal{A}$ is at most $r + 1$ times the running time of $\mathcal{R}$, which gives:

$$t_A = (r + 1) \cdot t_R \tag{38}$$

Now assume that the reduction $\mathcal{R}$ is allowed to rewind the forger to a previous state $\mathcal{S}$. Equivalently we assume that the reduction restarts the forger with the same random tape and provides the same input to the forger until the same state $\mathcal{S}$ is reached. If $\mathcal{R}$ restarts the forger at the $j-1$-th round, the reduction is then in the j-th round. We distinguish two cases: if the reduction sends the same public key and provides the same answer to the hash queries, the forger will make the same signature queries and forgery as in the $j-1$-th round. Therefore our simulation will make the same signature queries and forgery as in the $j-1$-th round. At the j-th round the reduction sees exactly the same transcript when interacting with the forger as when interacting with our simulation, except with probability:

$$\varepsilon_F \cdot \Pr[\mathbf{ans}_j \wedge \neg \mathbf{ans}'_j]$$

Otherwise if the reduction sends a different public key or provides a different answer to the hash queries, the forger makes signature queries for random messages and forge the

signature of a randomly chosen message, and so our simulation makes signature queries for random messages and forge the signature of a randomly chosen message. Consequently, at the j-th round the reduction sees exactly the same transcript when interacting with the forger as when interacting with our simulation, except with probability:

$$\varepsilon_F \cdot \Pr[\mathtt{ans}_j \wedge \neg \mathtt{ans}'_j]$$

Consequently, we obtain the same result as previously: $\mathcal{A}$ succeeds with probability at least ε_A given by equation (37).

G Security Definitions in the Standard Model

Definition 16. *A forger $\mathcal{F}$ is said to $(t, q_{sig}, \varepsilon)$-break the signature scheme* (`Gen`, `Sign`, `Verify`) *if after at most $q_{sig}(k)$ signature queries and $t(k)$ processing time, it outputs a forgery with probability at least $\varepsilon(k)$ for all $k \in \mathbb{N}$.*

Definition 17. *A signature scheme* (`Gen`, `Sign`, `Verify`) *is $(t, q_{sig}, \varepsilon)$-secure if there is no forger who $(t, q_{sig}, \varepsilon)$-breaks the scheme.*

Definition 18 (signature scheme with unique signature). *A signature scheme is said to be with unique signature if for any public key pk and any message M, there is a unique signature x such that $\mathtt{Verify}_{pk}(M, x) = 1$.*

Note that a signature scheme with unique signature is necessarily state-free: the signature of a message does not depend on previously signed messages.

We assume that the security of $(\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ is based on the hardness of the problem Π, so there exists a reduction from solving Π to breaking the signature scheme in the standard model.

Definition 19. *A reduction algorithm R in the standard model from solving $(Gen\Pi, D, S)$ to breaking* (`Gen`, `Sign`, `Verify`) *is a probabilistic algorithm taking as input an instance I and q_{sig}, where $I \leftarrow Gen\Pi(1^k)$, and outputting a solution z for I. The reduction algorithm interacts with a forger $\mathcal{F}$ for* (`Gen`, `Sign`, `Verify`) *which outputs a forgery after at most q_{sig} signature queries. The reduction algorithm answers the signature queries made by $\mathcal{F}$.*

Definition 20. *A reduction algorithm $\mathcal{R}$ is said to $(t_R, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduce solving Π to breaking the signature scheme* (`Gen`, `Sign`, `Verify`) *if after receiving an instance I such that $I \leftarrow Gen\Pi(1^k)$ and running any forger that $(t_F, q_{sig}, \varepsilon_F)$-breaks the signature scheme, the reduction $\mathcal{R}$ outputs a solution z for I with probability at least $\varepsilon_R(k)$ after at most $t_R(k)$ additional processing time for all $k \in \mathbb{N}$.*

H Proof of Theorem 6

The proof is very similar to the proof of theorem 5. The only difference is that there are no hash queries. Moreover, we replace in algorithm $\mathcal{A}$ the number of hash queries q_{hash} by the lower bound 2^ℓ on the size of the message space; instead of selecting q_{hash} distinct messages, we select 2^ℓ distinct messages $M_1, \ldots, M_{2^\ell}$. The rest of algorithm $\mathcal{A}$ is

the same, and the same analysis shows that from a reduction with running time of t_R, which succeeds with probability at least ε_R after running or rewinding at most r times a forger that breaks the signature scheme with probability at least ε_F, we can build an algorithm which (t_A, ε_A)-solves the problem Π, with:

$$t_A = (r+1) \cdot t_R$$
$$\varepsilon_A = \varepsilon_R - \varepsilon_F \cdot \frac{\exp(-1) \cdot r}{q_{sig}} \cdot \left(1 - \frac{q_{sig}}{2^\ell}\right)^{-1}$$

I A Variant of GHR's Scheme Provably Secure in the Standard Model

Let us consider Gennaro-Halevi-Rabin's signature scheme [11]. The public key is $N = p \cdot q$ and a random $y \in \mathbb{Z}_N^*$, where p and q are random $k/2$-bit primes and $(p-1)/2$ and $(q-1)/2$ are also primes. The private key is (p, q). The scheme uses a hash function H which outputs odd integers of length k_0 bits. To sign a message m, the signer obtains $e = H(m)$ and computes the signature σ as the e-th root of y modulo N, using p, q. To verify a signature σ, one computes $e = H(m)$ and checks that $\sigma^e = y \bmod N$.

The security of Gennaro-Halevi-Rabin's signature scheme is based on the hardness of the strong RSA problem.

Definition 21 (Strong RSA problem). *Given a randomly chosen RSA modulus N and a random element $s \in \mathbb{Z}_N^*$, find a pair (e, r) with $e > 1$ such that $r^e = s \bmod N$.*

In this section we illustrate theorem 6 with a variant of GHR's scheme provably secure in the standard model, with unique signature. The hash function H is replaced by an injective function Ψ which maps any string from $\{0,1\}^\ell$ to the set of prime integers, so that Ψ is easy to compute. Such a function is constructed in [11]. We obtain a signature scheme with unique signature provably secure in the standard model. However the scheme is provably secure for short messages only (say, less than 40 bits); this is due to the 2^ℓ factor in the time bound t_F of the forger. We denote by $t(\ell)$ the time necessary to compute Ψ.

Theorem 10. *Assume that the strong RSA problem is (t_I, ε_I)-hard. Then the previous GHR variant is $(t_F, q_{sig}, \varepsilon_F)$-secure, where:*

$$t_I = t_F + \mathtt{poly}\left(2^\ell, k, t(\ell)\right) \tag{39}$$
$$\varepsilon_I = \frac{\varepsilon_F}{q_{sig}} \cdot \left(1 - \frac{1}{q_{sig}+1}\right)^{q_{sig}+1} \tag{40}$$

Proof. Assume that there exists a forger $\mathcal{F}$ that $(t_F, q_{sig}, \varepsilon_F)$-breaks the signature scheme (`Gen`, `Sign`, `Verify`). We construct an algorithm $\mathcal{A}$ that solves the strong RSA problem using $\mathcal{F}$. $\mathcal{A}$ will answer the signature queries of the forger itself. The message space is $\{0,1\}^\ell$.

Algorithm for $\mathcal{A}$.
Input: (N, s) and (ℓ, q_{sig}), where $N \leftarrow RSA(1^k)$ and $s \xleftarrow{R} \mathbb{Z}_N^*$.
Output: (r, e) with $e > 1$ such that $r^e = s \bmod N$.

1. Let $E \leftarrow 1$.
2. For all messages $M_i \in \{0,1\}^\ell$, do the following:
   Flip a coin c_i with bias γ.
   $c_i = 0$ with probability γ and $c_i = 1$ with probability $1-\gamma$.
   If $c_i = 0$ then compute $E \leftarrow E \cdot \Psi(M_i)$.
3. Let $y \leftarrow s^E \bmod N$.
4. Send the public key (N, y) to $\mathcal{F}$.
5. If $\mathcal{F}$ makes a signature query for M_i:
   If $c_i = 0$ then return $s^{E/\Psi(M_i)} \bmod N$. Otherwise stop.
6. If $\mathcal{F}$ outputs a forgery (M_i, x):
   If $c_i = 0$ then stop.
   Otherwise $\Psi(M_i) \wedge E = 1$ so let $a, b \in \mathbb{Z}$ such that $a \cdot \Psi(M_i) + b \cdot E = 1$.
   Let $r \leftarrow x^b \cdot s^a \bmod N$ and $e \leftarrow \Psi(M_i)$ and output (r, e).

The probability that $\mathcal{A}$ answers to all the signature queries is greater than $\gamma^{q_{sig}}$. Eventually $\mathcal{F}$ outputs a forgery with probability ε_F which $\mathcal{A}$ can use with probability $1-\gamma$ to output (r, e). Consequently $\mathcal{A}$ outputs (r, e) with probability $\gamma^{q_{sig}} \cdot (1-\gamma) \cdot \varepsilon_F$, which is maximal for $\gamma = 1 - 1/(q_{sig}+1)$ and gives (40). □

J Proof of Theorem 7

Lemma 3. *Let $\mathcal{F}_0$ be a forger which $(t_F^0, q_{hash}^0, q_{sig}^0, \varepsilon_F^0)$-breaks $\mathtt{PSS0}[k_0, k_1]$. From $\mathcal{F}_0$ we can construct a forger $\mathcal{F}$ which $(t_F, q_{hash}, q_{sig}, \varepsilon_F)$-breaks $\mathtt{PSS}[k_0, k_1]$, with:*

$$q_{hash} = q_{hash}^0 \qquad \varepsilon_F = \varepsilon_F^0/2 \qquad q_{sig} = 2^{k_0+1} \cdot q_{sig}^0$$

$$t_F = t_F^0 + q_{sig} \cdot \mathcal{O}(k)$$

Proof. From $\mathcal{F}_0$ we construct a forger $\mathcal{F}$ for $\mathtt{PSS}[k_0, k_1]$. When the forger $\mathcal{F}_0$ makes a hash query, the forger $\mathcal{F}$ makes the same hash query and forwards the result to $\mathcal{F}_0$. When the forger $\mathcal{F}_0$ makes a signature query for a message M, the forger $\mathcal{F}$ makes signature queries for M until the random salt used to generate the signature is 0^{k_0}. Then it forwards the signature to $\mathcal{F}_0$. Eventually the forger $\mathcal{F}_0$ outputs a forgery for $\mathtt{PSS0}[k_0, k_1]$, which is also a forgery for $\mathtt{PSS}[k_0, k_1]$.

When $\mathcal{F}$ makes a signature query, the random salt used to generate the signature is equal to 0^{k_0} with probability 2^{-k_0}. Therefore $\mathcal{F}$ must perform on average 2^{k_0} signature queries for each signature query of $\mathcal{F}_0$. More precisely, let Y_i be the number of signature queries made by $\mathcal{F}$ for the i-th signature query of $\mathcal{F}_0$, and let Y be the total number of signature queries made by $\mathcal{F}$. Since $\mathcal{F}$ is limited to q_{sig} signature queries, the probability that all the signature queries of $\mathcal{F}_0$ are answered is $\Pr[Y \leq q_{sig}]$. In this case, the forger $\mathcal{F}_0$ outputs a forgery with probability at least ε_F^0. Therefore, the forger $\mathcal{F}$ outputs a forgery with probability at least $\varepsilon_F^0 \cdot \Pr[Y \leq q_{sig}]$.

The distribution of Y_i follows a geometric law of parameter $1 - 2^{-k_0}$:

$$\Pr[Y_i = j] = 2^{-k_0} \cdot (1 - 2^{-k_0})^{j-1} \quad \text{for } j \geq 1$$

The expectancy and variance of Y_i are given by:

$$\mathrm{E}[Y_i] = 2^{k_0} \qquad \mathrm{Var}[Y_i] = 2^{k_0} \cdot \left(2^{k_0} - 1\right)$$

We assume that $\mathcal{F}_0$ makes exactly q^0_{sig} signature queries[4]. Since Y is the sum of q^0_{sig} independent random variables, we obtain:

$$\mathrm{E}[Y] = 2^{k_0} \cdot q^0_{sig} \qquad \mathrm{Var}[Y] = q^0_{sig} \cdot 2^{k_0} \left(2^{k_0} - 1\right)$$

Then, using Chebyshev's inequality, we have for any δ:

$$\Pr\left[|Y - \mathrm{E}[Y]| \geq \delta\right] \leq \frac{\mathrm{Var}[Y]}{\delta^2}$$

and taking $\delta = E[Y]$, we obtain for $q^0_{sig} \geq 2$:

$$\Pr\left[Y \geq 2 \cdot E[Y]\right] \leq \frac{1}{q^0_{sig}} \leq \frac{1}{2}$$

If $q^0_{sig} = 1$, then $Y = Y_1$ and

$$\Pr[Y \geq 2 \cdot E[Y]] = \Pr[Y_1 \geq 2^{k_0+1}] = \left(1 - 2^{-k_0}\right)^{2^{k_0+1}-1}$$

Using the inequality $(1 - 1/x)^x \leq 1/2$ for $x \geq 1$, we obtain as previously:

$$\Pr[Y \geq 2 \cdot E[Y]] \leq \left(1 - 2^{-k_0}\right)^{2^{k_0}} \leq \frac{1}{2}$$

So letting

$$q_{sig} = 2^{k_0+1} \cdot q^0_{sig}$$

this gives $\Pr[Y \geq q_{sig}] \leq 1/2$ and thus $\Pr[Y \leq q_{sig}] \geq 1/2$. Consequently, the forger $\mathcal{F}$ outputs a forgery for $\mathtt{PSS}[k_0, k_1]$ with probability at least

$$\varepsilon_F = \varepsilon^0_F / 2$$

after at most q_{sig} signature queries. □

Lemma 4. *Let $\mathcal{R}$ be a reduction which $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces inverting RSA to breaking $\mathtt{PSS}[k_0, k_1]$. From $\mathcal{R}$ we can construct a reduction $\mathcal{R}_0$ which $(t^0_R, q^0_{hash}, q^0_{sig}, \varepsilon^0_F, \varepsilon^0_R)$-reduces inverting RSA to breaking $\mathtt{PSS0}[k_0, k_1]$, with:*

$$\begin{aligned} q_{sig} &= q^0_{sig} \cdot 2^{k_0+1} & q_{hash} &= q^0_{hash} \\ \varepsilon_F &= \varepsilon^0_F / 2 & \varepsilon_R &= \varepsilon^0_R \end{aligned} \tag{41}$$

$$t^0_R = t_R + q_{sig} \cdot \mathcal{O}(k) \tag{42}$$

[4] Otherwise we can simulate the remaining signature queries with previously queried messages. If $\mathcal{F}_0$ makes no signature queries, then $\mathcal{F}$ outputs a forgery with the same probability as $\mathcal{F}_0$.

Proof. Let $\mathcal{F}_0$ be a forger which $(t_F^0, q_{hash}^0, q_{sig}^0, \varepsilon_F^0)$-breaks $\texttt{PSS0}[k_0, k_1]$. Using the previous lemma we construct from $\mathcal{F}_0$ a forger $\mathcal{F}$ which $(t_F, q_{hash}, q_{sig}, \varepsilon_F)$-breaks $\texttt{PSS}[k_0, k_1]$, where q_{hash}, q_{sig} and ε_F are given by equation (41). Then from $\mathcal{F}$ using $\mathcal{R}$ we can invert RSA with probability at least ε_R.

Therefore, from $\mathcal{F}_0$ which $(t_F^0, q_{hash}^0, q_{sig}^0, \varepsilon_F^0)$-breaks $\texttt{PSS0}[k_0, k_1]$, and using $\mathcal{R}$, we can invert RSA with probability at least ε_R. Consequently, from $\mathcal{R}$ we can construct a reduction $\mathcal{R}_0$ which $(t_R^0, q_{hash}^0, q_{sig}^0, \varepsilon_F^0, \varepsilon_R^0)$-reduces inverting RSA to breaking $\texttt{PSS0}[k_0, k_1]$, where $\varepsilon_R^0 = \varepsilon_R$ and $t_R^0 = t_R + q_{sig} \cdot \mathcal{O}(k)$. □

Let $\mathcal{R}$ be a reduction which $(t_R, q_{hash}, q_{sig}, \varepsilon_F, \varepsilon_R)$-reduces inverting RSA to breaking $\texttt{PSS}[k_0, k_1]$. From lemma 4, we construct from $\mathcal{R}$ an algorithm $\mathcal{R}_0$ which $(t_R^0, q_{hash}^0, q_{sig}^0, \varepsilon_F^0, \varepsilon_R^0)$-reduces inverting RSA to breaking $\texttt{PSS0}[k_0, k_1]$, where t_R^0, q_{hash}^0, q_{sig}^0, ε_F^0 and ε_R^0 are given by equations (41) and (42). The reduction $\mathcal{R}$ can run or rewind the forger at most r times, so $\mathcal{R}_0$ runs or rewinds the forger at most r times. Then from $\mathcal{R}_0$ using theorem 5 we construct an algorithm $\mathcal{I}$ which (t_I, ε_I)-inverts RSA, with:

$$t_I = (r+1) \cdot t_R^0$$
$$\varepsilon_I = \varepsilon_R^0 - r \cdot \varepsilon_F^0 \cdot \frac{\exp(-1)}{q_{sig}^0} \cdot \left(1 - \frac{q_{sig}^0}{q_{hash}^0}\right)^{-1}$$

Using equations (41) and (42) with $q_{hash} \geq 2 \cdot q_{sig}$ and $\exp(-1) \leq 1/2$, we obtain:

$$r \cdot \varepsilon_F^0 \cdot \frac{\exp(-1)}{q_{sig}^0} \cdot \left(1 - \frac{q_{sig}^0}{q_{hash}^0}\right)^{-1} \leq r \cdot \varepsilon_F \cdot \frac{2^{k_0+2}}{q_{sig}}$$

which shows that the inverter succeeds with probability at least:

$$\varepsilon_R - r \cdot \varepsilon_F \cdot \frac{2^{k_0+2}}{q_{sig}}$$

and gives equations (14) and (15).

Universal Padding Schemes for RSA

Proceedings of Crypto 2002

Jean-Sébastien Coron, Marc Joye, David Naccache and Pascal Paillier

Gemplus Card International
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
{jean-sebastien.coron, marc.joye, david.naccache, pascal.paillier}@gemplus.com

Abstract. A common practice to encrypt with RSA is to first apply a padding scheme to the message and then to exponentiate the result with the public exponent; an example of this is OAEP. Similarly, the usual way of signing with RSA is to apply some padding scheme and then to exponentiate the result with the private exponent, as for example in PSS. Usually, the RSA modulus used for encrypting is different from the one used for signing. The goal of this paper is to simplify this common setting. First, we show that PSS can also be used for encryption, and gives an encryption scheme semantically secure against adaptive chosen-ciphertext attacks, in the random oracle model. As a result, PSS can be used indifferently for encryption or signature. Moreover, we show that PSS allows to safely use the same RSA key-pairs for both encryption and signature, in a concurrent manner. More generally, we show that using PSS the same set of keys can be used for both encryption and signature for any trapdoor partial-domain one-way permutation. The practical consequences of our result are important: PKIs and public-key implementations can be significantly simplified.

1 Introduction

A very common practice for encrypting a message m with RSA is to first apply a padding scheme μ, then raise $\mu(m)$ to the public exponent e. The ciphertext c is then:

$$c = \mu(m)^e \bmod N$$

Similarly, for signing a message m, the common practice consists again in first applying a padding scheme μ' then raising $\mu'(m)$ to the private exponent d. The signature s is then:

$$s = \mu'(m)^d \bmod N$$

For various reasons, it would be desirable to use the same padding scheme $\mu(m)$ for encryption and for signature: in this case, only one padding scheme needs to be implemented. Of course, the resulting padding scheme $\mu(m)$ should be provably secure for encryption and for signing. We say that a padding scheme is *universal* if it satisfies this property.

The strongest public-key encryption security notion was defined in [15] as *indistinguishability under an adaptive chosen ciphertext attack*. An adversary should not be able to distinguish between the encryption of two plaintexts, even if he can obtain the decryption of ciphertexts of his choice. For digital signature schemes, the strongest security notion was defined by Goldwasser, Micali and Rivest in [10], as *existential unforgeability under an adaptive chosen message attack*. This notion captures the property that an adversary cannot produce a valid signature, even after obtaining the signatures of (polynomially many) messages of his choice.

In this paper, we show that the padding scheme PSS [4], which is originally a provably secure padding scheme for producing signatures, can also be used as a provably secure encryption scheme. More precisely, we show that PSS offers indistinguishability under an adaptive chosen ciphertext attack, in the random oracle model, under the partial-domain one-wayness of the underlying permutation. Partial-domain one-wayness, introduced in [9], is a formally stronger assumption than one-wayness. However, for RSA, partial-domain one-wayness is equivalent to (full domain) one-wayness and therefore RSA-PSS encryption is provably secure under the sole assumption that RSA is one-way.

Generally, in a given application, the RSA modulus used for encrypting is different from the RSA modulus used for signing; our setting (and real-world PKIs) would be further simplified if one could use the same set of keys for both encryption and signature (see [11]). In this paper, we show that using PSS, the same keys can be safely used for encryption and for signature.

2 Public-Key Encryption

A public-key encryption scheme is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ where:

- $\mathcal{K}$ is a probabilistic key generation algorithm which returns random pairs of public and secret keys (pk, sk) depending on some security parameter k,
- $\mathcal{E}$ is a probabilistic encryption algorithm which takes as input a public key pk and a plaintext $M \in \mathcal{M}$, runs on a random tape $r \in \mathcal{R}$ and returns a ciphertext c. $\mathcal{M}$ and $\mathcal{R}$ stand for spaces in which messages and random strings are chosen respectively,
- $\mathcal{D}$ is a deterministic decryption algorithm which, given as input a secret key sk and a ciphertext c, returns the corresponding plaintext M, or Reject.

The strongest security notion for public-key encryption is the aforementioned notion of indistinguishability under an adaptive chosen ciphertext attack. An adversary should not be able to distinguish between the encryption of two plaintexts, even if he can obtain the decryption of ciphertexts of his choice. The attack scenario is the following:

1. The adversary $\mathcal{A}$ receives the public key pk with $(pk, sk) \leftarrow \mathcal{K}(1^\kappa)$.
2. $\mathcal{A}$ makes decryption queries for ciphertexts y of his choice.
3. $\mathcal{A}$ chooses two messages M_0 and M_1 of identical length, and receives the encryption c of M_b for a random unknown bit b.
4. $\mathcal{A}$ continues to make decryption queries. The only restriction is that the adversary cannot request the decryption of c.
5. $\mathcal{A}$ outputs a bit b', representing its "guess" on b.

The adversary's advantage is then defined as:

$$\mathsf{Adv}(\mathcal{A}) = |2 \cdot \Pr[b' = b] - 1|$$

An encryption scheme is said to be secure against adaptive chosen ciphertext attack (and denoted IND-CCA2) if the advantage of any polynomial-time bounded adversary is a negligible function of the security parameter. Usually, schemes are proven to be IND-CCA2 secure by exhibiting a polynomial reduction: if some adversary can break the IND-CCA2 security of the system, then the same adversary can be invoked (polynomially many times) to solve a related hard problem.

The random oracle model, introduced by Bellare and Rogaway in [2], is a theoretical framework in which any hash function is seen as an oracle which outputs a random value for each new query. Actually, a security proof in the random oracle model does not necessarily imply that a scheme is secure in the real world (see [7]). Nevertheless, it seems to be a good engineering principle to design a scheme so that it is provably secure in the random oracle model. Many encryption and signature schemes were proven to be secure in the random oracle model.

3 Encrypting with PSS-R

In this section we prove that given any trapdoor partially one-way permutation f, the encryption scheme defined by first applying PSS with message recovery (denoted PSS-R) and encrypting the result with f achieves the strongest security level for an encryption scheme, in the random oracle model.

3.1 The PSS-R Padding Scheme

PSS-R, defined in [4], is parameterized by the integers k, k_0 and k_1 and uses two hash functions:

$$H : \{0,1\}^{k-k_1} \to \{0,1\}^{k_1} \quad \text{and} \quad G : \{0,1\}^{k_1} \to \{0,1\}^{k-k_1}$$

PSS-R takes as input a $(k - k_0 - k_1)$-bit message M and a k_0-bit random integer r. As illustrated in figure 1, PSS-R outputs:

$$\mu(M, r) = \omega || s$$

where $||$ stands for concatenation, $\omega = H(M||r)$ and $s = G(\omega) \oplus (M\|r)$. Actually, in [4], $M\|r$ is used as the argument to H and $r\|M$ is used as the mask to xor with $G(\omega)$. Here for simplicity we use $M\|r$ in both places, but the same results apply either way.

3.2 The PSS-E Encryption Scheme

The new encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, that we denote PSS-E, is based on μ and a k-bit trapdoor permutation f.

- $\mathcal{K}$ generates the public key f and the secret key f^{-1}.
- $\mathcal{E}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random $r \in \{0,1\}^{k_0}$, the encryption algorithm outputs the ciphertext:

$$c = f(\mu(M, r))$$

- $\mathcal{D}(c)$: the decryption algorithm recovers $(\omega, s) = f^{-1}(c)$ and then $M\|r = G(\omega) \oplus s$. If $\omega = H(M||r)$, the algorithm returns M, otherwise it returns Reject.

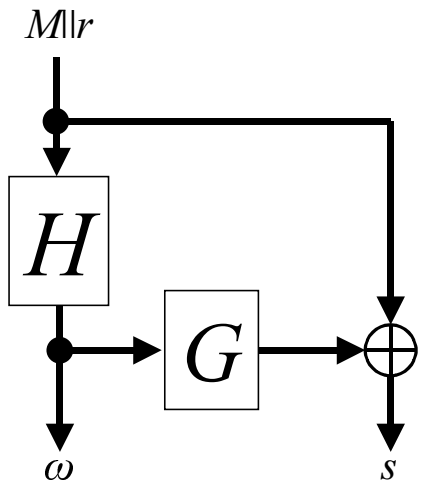

Fig. 1. The PSS-R padding scheme

3.3 The Underlying Problem

The security of PSS-E is based on the difficulty of inverting f without knowing f^{-1}. As in [9], we use two additional related problems: the partial-domain one-wayness and the set partial-domain one-wayness of f:

- (τ, ε)**-one-wayness of** f, means that for any adversary $\mathcal{A}$ who wishes to recover the full pre-image (ω, s) of $f(\omega, s)$ in time less than τ, $\mathcal{A}$'s success probability $\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A})$ is upper-bounded by ε:

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr_{\omega,s}[\mathcal{A}(f(\omega, s)) = (\omega, s)] < \varepsilon$$

- (τ, ε)**-partial-domain one-wayness of** f, means that for any adversary $\mathcal{A}$ who wishes to recover the partial pre-image ω of $f(\omega, s)$ in time less than τ, $\mathcal{A}$'s success probability $\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A})$ is upper-bounded by ε:

$$\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A}) = \Pr_{\omega,s}[\mathcal{A}(f(\omega, s)) = \omega] < \varepsilon$$

- $(\ell, \tau, \varepsilon)$**-set partial-domain one-wayness of** f, means that for any adversary $\mathcal{A}$ who wishes to output a set of ℓ elements which contains the partial pre-image ω of $f(\omega, s)$, in time less than τ, $\mathcal{A}$'s success probability $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A})$ is upper-bounded by ε:

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A}) = \Pr_{\omega,s}[\omega \in \mathcal{A}(f(\omega, s))] < \varepsilon$$

As in [9], we denote by $\mathsf{Succ}^{\mathsf{ow}}(\tau)$, (resp. $\mathsf{Succ}^{\mathsf{pd-ow}}(\tau)$ and $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\ell, \tau)$) the maximal probability $\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A})$, (resp. $\mathsf{Succ}^{\mathsf{pd-ow}}(\mathcal{A})$ and $\mathsf{Succ}^{\mathsf{s-pd-ow}}(\mathcal{A})$), over all adversaries whose running times are less than τ. For any τ and $\ell \geq 1$, we have:

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(\ell, \tau) \geq \mathsf{Succ}^{\mathsf{pd-ow}}(\tau) \geq \mathsf{Succ}^{\mathsf{ow}}(\tau)$$

Moreover, by randomly selecting any element in the set returned by the adversary against the set partial-domain one-wayness, one can break the partial-domain one-wayness with probability $1/\ell$, which gives:

$$\mathsf{Succ}^{\mathsf{pd-ow}}(\tau) \geq \mathsf{Succ}^{\mathsf{s-pd-ow}}(\ell, \tau)/\ell \tag{1}$$

We will see in Section 5 that for RSA, the three problems are polynomially equivalent.

3.4 Security of PSS-E

The following theorem shows that PSS-E is semantically secure under adaptive chosen ciphertext attacks, in the random oracle model, assuming that the underlying permutation is partially one-way.

Theorem 1. *Let $\mathcal{A}$ be a CCA2-adversary against the semantic security of PSS-E $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, with advantage ε and running time t, making q_D, q_H and q_G queries to the decryption oracle and the hash functions H and G, respectively. Then:*

$$\mathsf{Succ}^{\mathsf{pd-ow}}(t') \geq \frac{1}{q_H + q_G} \cdot \left(\varepsilon - q_H 2^{-k_0} - q_D 2^{-k_1}\right)$$

where $t' \leq t + q_H \cdot T_f$, and T_f denotes the time complexity of f.

The theorem follows from inequality (1) and the following lemma:

Lemma 1. *Using the notations introduced in theorem 1, we have:*

$$\mathsf{Succ}^{\mathsf{s-pd-ow}}(q_H + q_G, t') \geq \varepsilon - q_H \cdot 2^{-k_0} - q_D \cdot 2^{-k_1} \tag{2}$$

Proof. We describe a reduction $\mathcal{B}$ which using $\mathcal{A}$, constructs an adversary against the set partial-domain one-wayness of f. We start with a top-level description of the reduction and then show how to simulate the random oracles G, H and the decryption oracle D. Eventually we compute the success probability of $\mathcal{B}$.

Top-level description of the reduction $\mathcal{B}$:

1. $\mathcal{B}$ is given a function f and $c^* = f(\omega^*, s^*)$, for a random ω^* and s^*. $\mathcal{B}$'s goal is to output a list which contains the partial pre-image ω^* of c^*.

2. $\mathcal{B}$ runs $\mathcal{A}$ with f and gets $\{M_0, M_1\}$. It chooses a random bit b and gives c^* as a ciphertext for M_b. $\mathcal{B}$ simulates the decryption oracle H, D and G as described below.

3. $\mathcal{B}$ receives from $\mathcal{A}$ the answer b' and outputs the list of queries asked to G.

Simulation of the random oracles G, H and D.

The simulation of G and H is very simple: a random answer is returned for each new query of G and H. Moreover, when ω is the answer of a query to H, we simulate a query for ω to G, so that $G(\omega)$ is defined.

On query c to the decryption oracle, the reduction $\mathcal{B}$ looks at each query $M'||r'$ to H and computes:

$$\omega' = H(M'||r') \text{ and } s' = G(\omega') \oplus (M'\|r')$$

Then if $c = f(\omega', s')$ the reduction $\mathcal{B}$ returns M'. Otherwise, the reduction outputs Reject.

Analysis:

Since $c^* = f(\omega^*, s^*)$ is the ciphertext corresponding to M_b, we have the following constraint for the random oracles G and H:

$$H(M_b \| r^*) = \omega^* \quad \text{and} \quad G(\omega^*) = s^* \oplus (M_b \| r^*) \tag{3}$$

We denote by AskG the event: "ω^* has been asked to G" and by AskH the event: "there exists M' such that $M'||r^*$ has been queried to H".

If ω^* was never queried to G, then $G(\omega^*)$ is undefined and r^* is then a uniformly distributed random variable. Therefore the probability that there exists M' such that (M', r^*) has been asked to H is at most $q_H \cdot 2^{-k_0}$. This gives:

$$\Pr[\mathsf{AskH} | \neg \mathsf{AskG}] \leq q_H \cdot 2^{-k_0} \tag{4}$$

Our simulation of D can only fail by rejecting a valid ciphertext. We denote by DBad this event. Letting $c = f(\omega, s)$ be the ciphertext queried to D and

$$M \| r = G(\omega) \oplus s$$

we reject a valid ciphertext if $H(M||r) = \omega$ while $M||r$ was never queried to H. However, if $M\|r$ was never queried to H, then $H(M\|r)$ is randomly defined. Namely if the decryption query occured before c^* was sent to the adversary, then constraint (3) does not apply and $H(M\|r)$ is randomly defined. Otherwise, if the decryption query occured after c^* was sent to the adversary, then $c \neq c^*$ implies $(M, r) \neq (M_b, r^*)$ and $H(M\|r)$ is still randomly defined. In both cases the probability that $H(M, r) = \omega$ is then 2^{-k_1}, which gives:

$$\Pr[\mathsf{DBad}] \leq q_D \cdot 2^{-k_1} \tag{5}$$

Let us denote by Bad the event: "ω^* has been queried to G or (M', r^*) has been queried to H for some M' or the simulation of D has failed". Formally:

$$\mathsf{Bad} = \mathsf{AskG} \vee \mathsf{AskH} \vee \mathsf{DBad} \tag{6}$$

Let us denote by S the event: "the adversary outputs the correct value for b, *i.e.*, $b = b'$". Conditioned on $\neg$Bad, our simulations of G, H and D are independent of b, and therefore $\mathcal{A}$'s view is independent of b as well. This gives:

$$\Pr[\mathsf{S} | \neg \mathsf{Bad}] = \frac{1}{2} \tag{7}$$

Moreover, conditioned on $\neg$Bad, the adversary's view is the same as when interacting with (perfect) random and decryption oracles, which gives:

$$\Pr[\mathsf{S} \wedge \neg \mathsf{Bad}] \geq \frac{1}{2} + \frac{\varepsilon}{2} - \Pr[\mathsf{Bad}] \tag{8}$$

From (7) we obtain

$$\Pr[\mathsf{S} \wedge \neg \mathsf{Bad}] = \Pr[\mathsf{S} | \neg \mathsf{Bad}] \cdot \Pr[\neg \mathsf{Bad}] = \frac{1}{2}(1 - \Pr[\mathsf{Bad}])$$

which gives using (8):

$$\Pr[\mathsf{Bad}] \geq \varepsilon \tag{9}$$

From (6) we have:

$$\begin{aligned}\Pr[\mathsf{Bad}] &\leq \Pr[\mathsf{AskG} \vee \mathsf{AskH}] + \Pr[\mathsf{DBad}] \\ &\leq \Pr[\mathsf{AskG}] + \Pr[\mathsf{AskH} \wedge \neg\mathsf{AskG}] + \Pr[\mathsf{DBad}] \\ &\leq \Pr[\mathsf{AskG}] + \Pr[\mathsf{AskH}|\neg\mathsf{AskG}] + \Pr[\mathsf{DBad}]\end{aligned}$$

which yields using (4), (5) and (9):

$$\Pr[\mathsf{AskG}] \geq \varepsilon - q_H \cdot 2^{-k_0} - q_D \cdot 2^{-k_1}$$

and hence (2) holds. This terminates the proof of lemma 1. □

4 Signing and Encrypting with the Same Public-Key

In this section we show that when using PSS, the same public key can be used for encryption and signature in a concurrent manner. For RSA, this means that the same pair (N, e) can be used for both operations. In other words, when Alice sends a message to Bob, she encrypts it using Bob's public key (N, e); Bob decrypts it using the corresponding private key (N, d). To sign a message M, Bob will use the *same* private key (N, d). As usual, anybody can verify Bob's signature using his public pair (N, e).

Although provably secure (as we will see hereafter), this is contrary to the folklore recommendation that signature and encryption keys should be distinct. This recommendation may prove useful is some cases; this is particularly true when a flaw has been found in the encryption scheme or in the signature scheme. In our case, we will prove that when using the PSS-R padding scheme, a decryption oracle does not help the attacker in forging signatures, and a signing oracle does not help the attacker in gaining information about the plaintext corresponding to a ciphertext.

Nevertheless, we advise to be very careful when implementing systems using the same keys for encrypting and signing. For example, if there are some implementation errors in a decryption server (see for example [13]), then an attacker could use this server to create forgeries.

4.1 Signature Schemes and Their Security

Definition 1 (signature scheme). *A signature scheme* $(\mathtt{Gen}, \mathtt{Sign}, \mathtt{Verify})$ *is defined as follows:*

- The key generation algorithm `Gen` *is a probabilistic algorithm which given* 1^k*, outputs a pair of matching public and private keys,* (pk, sk)*.*

- The signing algorithm `Sign` *takes the message* M *to be signed, the public key* pk *and the private key* sk*, and returns a signature* $x = \mathtt{Sign}_{sk}(M)$*. The signing algorithm may be probabilistic.*

- The verification algorithm `Verify` *takes a message* M*, a candidate signature* x' *and* pk*. It returns a bit* $\mathtt{Verify}_{pk}(M, x')$*, equal to one if the signature is accepted, and zero otherwise. We require that if* $x \leftarrow \mathtt{Sign}_{sk}(M)$*, then* $\mathtt{Verify}_{pk}(M, x) = 1$*.*

In the *existential unforgeability under an adaptive chosen message attack* scenario, the forger can dynamically obtain signatures of messages of his choice and attempts to output a valid forgery. A *valid forgery* is a message/signature pair (M, x) such that $\texttt{Verify}_{pk}(M, x) = 1$ whereas the signature of M was never requested by the forger.

4.2 The PSS-ES Encryption and Signature Scheme

The PSS-ES encryption and signature scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{S}, \mathcal{V})$ is based on PSS-R and a k-bit trapdoor permutation f. As for the PSS-R signature scheme, the signature scheme in PSS-ES is with message recovery: this means that the message is recovered when verifying the signature. In this case, only messages of fixed length $k - k_0 - k_1$ can be signed. To sign messages M of arbitrary length, it suffices to apply a collision-free hash function to M prior to signing.

- $\mathcal{K}$ generates the public key f and the secret key f^{-1}.
- $\mathcal{E}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random value $r \in \{0,1\}^{k_0}$, the encryption algorithm computes the ciphertext:

$$c = f(\mu(M, r))$$

- $\mathcal{D}(c)$: the encryption algorithm recovers $(\omega, s) = f^{-1}(c)$ and computes

$$M \| r = G(\omega) \oplus s$$

If $\omega = H(M||r)$, the algorithm returns M, otherwise it returns Reject.

- $\mathcal{S}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random value $r \in \{0,1\}^{k_0}$, the signing algorithm computes the signature:

$$\sigma = f^{-1}(\mu(M, r))$$

- $\mathcal{V}(\sigma)$: given the signature σ, the verification algorithm recovers $(\omega, s) = f(\sigma)$ and computes:

$$M \| r = G(\omega) \oplus s$$

If $\omega = H(M\|r)$, the algorithm accepts the signature and returns M. Otherwise, the algorithm returns Reject.

4.3 Semantic Security

We must ensure that an adversary is still unable to distinguish between the encryption of two messages, even if he can obtain the decryption of ciphertexts of his choice, and the signature of messages of his choice. The attack scenario is consequently the same as previously, except that the adversary can also obtain the signature of messages he wants.

The following theorem, whose proof is given in Appendix A, shows that PSS-ES is semantically secure under adaptive chosen ciphertext attacks, in the random oracle model, assuming that the underlying permutation is partial domain one-way.

Theorem 2. *Let $\mathcal{A}$ be an adversary against the semantic security of PSS-ES, with success probability ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle, and the hash functions H and G, respectively. Then,* $\mathsf{Succ}^{\mathsf{pd-ow}}(t')$ *is greater than:*

$$\frac{1}{q_H + q_G + q_{sig}} \left(\varepsilon - (q_H + q_{sig}) \cdot 2^{-k_0} - q_D 2^{-k_1} - (q_H + q_{sig})^2 \cdot 2^{-k_1} \right)$$

where $t' \leq t + (q_H + q_{sig}) \cdot T_f$, and T_f denotes the time complexity of f.

4.4 Unforgeability

For signature schemes, the strongest security notion is the previously introduced existential unforgeability under an adaptive chosen message attack. An attacker cannot produce a valid signature, even after obtaining the signature of (polynomially many) messages of his choice. Here the adversary can also also obtain the decryption of ciphertexts of his choice under the same public-key. Consequently, the attack scenario is the following:

1. The adversary $\mathcal{A}$ receives the public key pk with $(pk, sk) \leftarrow \mathcal{K}(1^\kappa)$.
2. $\mathcal{A}$ makes signature queries for messages M of his choice. Additionally, he makes decryption queries for ciphertexts y of his choice.
3. $\mathcal{A}$ outputs the signature of a message M' which was not queried for signature before.

An encryption-signature scheme is said to be secure against chosen-message attacks if for any polynomial-time bounded adversary, the probability to output a forgery is negligible.

The following theorem shows that PSS-ES is secure against an adaptive chosen message attack. The proof is similar to the security proof of PSS [4] and is given in Appendix B.

Theorem 3. *Let $\mathcal{A}$ be an adversary against the unforgeability of PSS-ES, with success probability ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle, and the hash oracles H and G, respectively. Then* $\mathsf{Succ}^{\mathsf{ow}}(t')$ *is greater than:*

$$\frac{1}{q_H} \left(\varepsilon - ((q_H + q_{sig})^2 + q_D + 1) \cdot 2^{-k_1} \right) \tag{10}$$

where $t' \leq t + (q_H + q_{sig}) \cdot T_f$, and T_f denotes the time complexity of f.

5 Application to RSA

5.1 The RSA Cryptosystem

The RSA cryptosystem, invented by Rivest, Shamir and Adleman [16], is the most widely used cryptosystem today. In this section, we show that by virtue of RSA's homomorphic properties, the partial-domain one-wayness of RSA is equivalent to the one-wayness of RSA. This enables to prove that the encryption scheme RSA-PSS-E and the encryption and signature scheme RSA-PSS-ES are semantically secure against chosen ciphertext attacks, in the random oracle model, assuming that inverting RSA is hard.

Definition 2 (The RSA Primitive). *The RSA primitive is a family of trapdoor permutations, specified by:*

- The RSA generator $\mathcal{RSA}$, which on input 1^k, randomly selects two distinct $k/2$-bit primes p and q and computes the modulus $N = p \cdot q$. It randomly picks an encryption exponent $e \in \mathbb{Z}^*_{\phi(N)}$, computes the corresponding decryption exponent $d = e^{-1} \bmod \phi(N)$ and returns (N, e, d).

- The encryption function $f : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ defined by $f(x) = x^e \bmod N$.

- The decryption function $f^{-1} : \mathbb{Z}^*_N \to \mathbb{Z}^*_N$ defined by $f^{-1}(y) = y^d \bmod N$.

In the following, we state our result in terms of the RSA primitive with a randomly chosen public exponent. The same results apply to the common practice of choosing a small public exponent. Actually, using Coppersmith's algorithm [8] as in [17] for OAEP [3], it would be possible to obtain tigther bounds for a small public exponent.

5.2 Partial-Domain One-Wayness of RSA

The following lemma shows that the partial-domain one-wayness of RSA is equivalent to the one-wayness of RSA. This is a generalization of the result that appears in [9] for OAEP and in [5] for SAEP$^+$, wherein the size of the partial pre-image is always greater than half the size of the modulus. [9] relies upon lattice reduction techniques for lattices of dimension 2. Here the partial pre-image can be smaller than half the size of the modulus (*e.g* a 160-bit pre-image for a 1024-bit modulus), so we must consider lattices of higher dimension. The extension was announced in [9] and [5], even if the proper estimates were not worked out.

The technique goes as follows. Given $y = x^e \bmod N$, we must find x. We obtain the least significant bits of $x \cdot \alpha_i \bmod N$ for random integers $\alpha_i \in \mathbb{Z}_N$, by querying for the partial pre-image of $y_i = y \cdot (\alpha_i)^e \bmod N$. Finding x from the least significant bits of the $x \cdot \alpha_i \bmod N$ is a Hidden Number Problem modulo N. We use an algorithm similar to [6] to efficiently recover x.

Lemma 2. *Let $\mathcal{A}$ be an algorithm that on input y, outputs a q-set containing the k_1 most significant bits of $y^d \bmod N$, within time bound t, with probability ε, where $2^{k-1} \leq N < 2^k$, $k_1 \geq 64$ and $k/(k_1)^2 \leq 2^{-6}$. Then there exists an algorithm $\mathcal{B}$ that solves the RSA problem with success probability ε' within time bound t', where:*

$$\varepsilon' \geq \varepsilon \cdot (\varepsilon^{n-1} - 2^{-k/8}) \tag{11}$$

$$t' \leq n \cdot t + q^n \cdot \mathtt{poly}(k)$$

$$n = \left\lceil \frac{5k}{4k_1} \right\rceil$$

Proof. Algorithm $\mathcal{B}$ receives y as input and must output $y^d \bmod N$. It generates the integers $\alpha_i \in \mathbb{Z}_N$ at random for $1 \leq i \leq n-1$, where n is an integer which will be determined later.

Let $y_0 = y$ and $y_i = y \cdot (\alpha_i)^e \bmod N$ for $1 \leq i \leq n-1$. We write for $0 \leq i \leq n-1$:

$$(y_i)^d = \omega_i \cdot 2^{k-k_1} + s_i \bmod N$$

where $0 \leq s_i < 2^{k-k_1}$. Letting $k_2 = k - k_1$, we obtain for $1 \leq i \leq n-1$:

$$\alpha_i \cdot (\omega_0 \cdot 2^{k_2} + s_0) = \omega_i \cdot 2^{k_2} + s_i \bmod N$$

Therefore letting $c_i = 2^{k_2} \cdot (\alpha_i \cdot \omega_0 - \omega_i) \bmod N$, we obtain the following system of $n-1$ equations in the n unknown s_i:

$$\mathcal{S}: \quad s_i - \alpha_i \cdot s_0 = c_i \bmod N \quad \text{for } 1 \leq i \leq n-1 \tag{12}$$

The following lemma, whose proof is given in appendix C, shows that given the c_i and α_i, the s_i can be recovered in time polynomial in k. We denote by:

$$\|\boldsymbol{x}\|_\infty = \max |x_i|$$

the infinite norm of vector $\boldsymbol{x}$.

Lemma 3. *If the previous set $\mathcal{S}$ of equations has a solution $\boldsymbol{s} = (s_0, \ldots, s_{n-1})$ such that $\|\boldsymbol{s}\|_\infty < 2^{k_2}$, then for all values of $\boldsymbol{\alpha}$, except a fraction:*

$$\frac{2^{n \cdot (k_2+n+2)}}{N^{n-1}} \tag{13}$$

of them, this solution is unique and can be computed in time polynomial in n and in the size of N.

Consequently, algorithm $\mathcal{B}$ runs n times algorithm $\mathcal{A}$ with input y_i. It obtains n sets of q integers, each set containing ω_i. Then it applies q^n times the algorithm of lemma 3, one execution of the algorithm enabling to recover the s_i, with probability:

$$\varepsilon' \geq \varepsilon \cdot \left(\varepsilon^{n-1} - \frac{2^{n \cdot (k_2+n+2)}}{N^{n-1}} \right)$$

In appendix D we show that taking $n = \lceil 5k/(4k_1) \rceil$ we obtain:

$$\frac{2^{n \cdot (k_2+n+2)}}{N^{n-1}} \leq 2^{-k/8} \tag{14}$$

which gives (11). □

5.3 RSA-PSS-E and RSA-PSS-ES

The RSA-PSS-E encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ based on the PSS-R padding μ with parameters k, k_0, and k_1 is defined as follows:

- $\mathcal{K}$ generates a $(k+1)$-bit RSA modulus and exponents e and d. The public key is (N, e) and the private key is (N, d).
- $\mathcal{E}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random $r \in \{0,1\}^{k_0}$, the encryption algorithm outputs the ciphertext:

$$c = (\mu(M, r))^e \bmod N$$

- $\mathcal{D}(c)$: the decryption algorithm recovers $x = c^d \bmod N$. It returns Reject if the most significant bit of x is not zero. It writes x as $0\|\omega\|s$ where ω is a k_1-bit string and s is a $k - k_1$ bit string. It writes $M\|r = G(\omega) \oplus s$. If $\omega = H(M||r)$, the algorithm returns M, otherwise it returns Reject.

The RSA-PSS-ES encryption and signature scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{S}, \mathcal{V})$ is defined as follows:

- $\mathcal{K}$, $\mathcal{E}(M, r)$ and $\mathcal{D}(c)$ are identical to RSA-PSS-E.

- $\mathcal{S}(M, r)$: given a message $M \in \{0,1\}^{k-k_0-k_1}$ and a random value $r \in \{0,1\}^{k_0}$, the signing algorithm computes the signature:

$$\sigma = \mu(M, r)^d \bmod N$$

- $\mathcal{V}(\sigma)$: given the signature σ, the verification algorithm recovers $x = \sigma^e \bmod N$. It returns Reject if the most significant bit of x is not zero. It writes x as $0\|\omega\|s$ where ω is a k_1-bit string and s is a $k - k_1$ bit string. It writes $M\|r = G(\omega) \oplus s$. If $\omega = H(M||r)$, the algorithm accepts the signature and returns M, otherwise it returns Reject.

5.4 Security of RSA-PSS-E and RSA-PSS-ES

Combining lemma 1 and lemma 2, we obtain the following theorem which shows that the encryption scheme RSA-PSS-E is provably secure in the random oracle model, assuming that inverting RSA is hard.

Theorem 4. *Let $\mathcal{A}$ be a CCA2-adversary against the semantic security of the RSA-PSS-E scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, with advantage ε and running time t, making q_D, q_H and q_G queries to the decryption oracle and the hash function H and G, respectively. We assume that $k_1 \geq 64$ and $k/(k_1)^2 \leq 2^{-6}$. Then we can invert RSA with probability ε' greater than:*

$$\varepsilon' \geq \left(\varepsilon - q_H \cdot 2^{-k_0} - q_D 2^{-k_1}\right)^n - 2^{-k/8}$$

within time bound $t' \leq n \cdot t + (q_H + q_G)^n \cdot \mathtt{poly}(k) + n \cdot q_H \cdot \mathcal{O}(k^3)$, where $n = \lceil 5k/(4k_1) \rceil$.

We obtain a similar theorem for the semantic security of the RSA-PSS-ES encryption and signature scheme (from Lemma 2 and Lemma 4 in appendix A).

Theorem 5. *Let $\mathcal{A}$ be a CCA2-adversary against the semantic security of the RSA-PSS-ES scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{S}, \mathcal{V})$, with advantage ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle and the hash function H and G, respectively. Provided that $k_1 \geq 64$ and $k/(k_1)^2 \leq 2^{-6}$, RSA can be inverted with probability ε' greater than:*

$$\varepsilon' \geq \left(\varepsilon - (q_H + q_{sig}) \cdot 2^{-k_0} - (q_D + (q_H + q_{sig})^2) \cdot 2^{-k_1}\right)^n - 2^{-k/8}$$

within time bound $t' \leq n \cdot t + (q_H + q_G + q_{sig})^n \cdot \mathtt{poly}(k)$, where $n = \lceil 5k/(4k_1) \rceil$.

For the unforgeability of the RSA-PSS-ES encryption and signature scheme, we obtain a better security bound than the general result of Theorem 3, by relying upon the homomorphic properties of RSA. The proof of the following theorem is similar to the security proof of PSS in [4] and is given in appendix E.

Theorem 6. *Let $\mathcal{A}$ be an adversary against the unforgeability of the PSS-ES scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{S}, \mathcal{V})$, with success probability ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle, and the hash functions H and G, respectively. Then RSA can be inverted with probability ε' greater than:*

$$\varepsilon' \geq \varepsilon - \left((q_H + q_{sig})^2 + q_D + 1\right) \cdot (2^{-k_0} + 2^{-k_1}) \tag{15}$$

within time bound $t' \leq t + (q_H + q_{sig}) \cdot \mathcal{O}(k^3)$.

Note that as for OAEP [9], the security proof for encrypting with PSS is far from being tight. This means that it does not provide a meaningful security result for a moderate size modulus (*e.g.*, 1024 bits). For the security proof to be meaningful in practice, we recommend to take $k_1 \geq k/2$ and to use a larger modulus (*e.g.*, 2048 bits).

6 Conclusion

In all existing PKIs different padding formats are used for encrypting and signing; moreover, it is recommended to use different keys for encrypting and signing. In this paper we have proved that the PSS padding scheme used in PKCS#1 v.2.1 [14] and IEEE P1363 [12] can be safely used for encryption as well. We have also proved that the same key pair can be safely used for both signature and encryption. The practical consequences of this are significant: besides halving the number of keys in security systems and simplifying their architecture, our observation allows resource-constrained devices such as smart cards to use the same code for implementing both signature and encryption.

Acknowledgements

We wish to thank Jacques Stern for pointing out an error in an earlier version of this paper, and the anonymous referees of Crypto 2002 for their useful comments.

References

1. L. Babai, *On Lovász' lattice reduction and the nearest lattice point problem*, Combinatorica vol. 6, no. 1, pp. 1–13, 1986.
2. M. Bellare and P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Commmunications Security (ACM-CCS), pp. 62–73, 1993.
3. M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption*, Proceedings of Eurocrypt'94, Lecture Notes in Computer Science vol. 950, Springer-Verlag, pp. 92–111, 1994.
4. M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*. Proceedings of Eurocrypt'96, Lecture Notes in Computer Science vol. 1070, Springer-Verlag, pp. 399-416, 1996.
5. D. Boneh, *Simplified OAEP for the RSA and Rabin Functions*, Proceedings of Crypto 2001, Lecture Notes in Computer Science vol. 2139, Springer-Verlag, pp. 275–291, 2001.ß

6. D. Boneh, R. Venkatesan, *Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes*, Proceedings of Crypto'96, Lecture Notes in Computer Science vol. 1109, Springer-Verlag, pp. 129–142, 1996. ß
7. R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology, revisited*, Proceedings of the 30-th Annual ACM Symposium on the Theory of Computing, pp. 209–218, 1998.
8. D. Coppersmith, *Finding a small root of a univariate modular equation*, Proceedings of Eurocrypt'96, Lecture Notes in Computer Science vol. 1070, pp. 155–165, 1996.
9. E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern, *RSA-OAEP is secure under the RSA assumption*, Proceedings of Crypto' 2001, Lecture Notes in Computer Science vol. 2139, Springer-Verlag, pp. 260-274, 2001.
10. S. Goldwasser, S. Micali and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of computing, 17(2), pp. 281–308, 1988.
11. S. Haber and B. Pinkas, *Securely Combining Public Key Cryptosystems*, Proceedings of the 8-th ACM Computer and Security Conference, pp. 215–224, 2001.
12. IEEE P1363a, *Standard Specifications For Public Key Cryptography: Additional Techniques*, `http://www.manta.ieee.org/groups/1363`
13. J. Manger, *A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS#1 v2.0*, Proceedings of Crypto 2001, Lecture Notes in Computer Science vol. 2139, Springer-Verlag, pp. 230–238, 2001.
14. PKCS #1 v2.1, *RSA Cryptography Standard (draft)*, `http://www.rsasecurity.com/rsalabs/pkcs`.
15. C.Rackoff and D. Simon, *Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack*. Adances in Cryptology, Crypto'91, Lecture Notes in Computer Science vol. 576, pages 433-444, 1992.
16. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, CACM 21(2), pp. 120–126, 1978.
17. V. Shoup, *OAEP Reconsidered*, Proceedings of Crypto 2001, Lecture Notes in Computer Science vol. 2139, Springer-Verlag, pp. 239–259, 2001.

A Proof of Theorem 2

The theorem follows from inequality (1) and the following lemma.

Lemma 4. *Let $\mathcal{A}$ be an adversary against the semantic security of PSS-ES, with success probability ε and running time t, making q_D, q_{sig}, q_H and q_G queries to the decryption oracle, the signing oracle, and the hash functions H and G, respectively. Then, the success probability $\mathsf{Succ}^{\mathsf{s-pd-ow}}(q_G, t')$ is greater than:*

$$\varepsilon - (q_H + q_{sig}) \cdot 2^{-k_0} - q_D 2^{-k_1} - (q_H + q_{sig})^2 \cdot 2^{-k_1}$$

where $t' \leq t + (q_H + q_{sig}) \cdot T_f$, and T_f denotes the time complexity of f.

Proof. The proof is very similar to the proof of lemma 1. The top-level description of the reduction $\mathcal{B}$ is the same and the simulation of the decryption oracle is the same. However, oracles H and G are simulated differently. Instead of simulating H and G so that $\mu(M,r) = y$ is a random integer, we simulate H and G so that $\mu(M,r) = f(x)$ for a known random x, which allows to answer the signature query for M.

Simulation of oracles G and H and signing oracle:

When receiving the query $M\|r$ to H, we generate a random $x \in \{0,1\}^k$ and compute $y = f(x)$. We denote $y = \omega\|s$. If ω never appeared before, we let $G(\omega) = s \oplus (M\|r)$ and return ω, otherwise we abort.

When receiving a query ω for G, if $G(\omega)$ has already been defined, we return $G(\omega)$, otherwise we return a random $(k-k_1)$-bit integer.

When we receive a signature query for M, we generate a random k_0-bit integer r. If $M\|r$ was queried to H before, we know ω, s, y and x such that:

$$H(M\|r) = \omega \quad \text{and} \quad G(\omega) = s \oplus (M\|r) \quad \text{and} \quad y = f(x) = \omega\|s$$

so we return the corresponding signature x. If $M\|r$ was never queried before, we simulate an H-query for $M\|r$ as previously: we pick a random $x \in \{0,1\}^k$ and compute $y = f(x)$. We denote $y = \omega\|s$. If ω never appeared before, we let $H(M\|r) = \omega$, $G(\omega) = s \oplus (M\|r)$ and return the signature x, otherwise we abort.

Analysis

As in lemma 1, we denote by AskG the event: "ω^* has been asked to G" and by AskH the event: "there exists M' such that $M'||r^*$ has been queried to H"; we denote by DBad the event: "a valid ciphertext has been rejected by our simulation of the decryption oracle D". Moreover, we denote by SBad the event: "the reduction aborts when answering a H-oracle query or a signature query". As previously, we have:

$$\Pr[\mathsf{AskH}|\neg\mathsf{AskG}] \leq (q_H + q_{sig}) \cdot 2^{-k_0}$$

and

$$\Pr[\mathsf{DBad}] \leq q_D \cdot 2^{-k_1}$$

When answering an H-oracle query or a signature query, the integer ω which is generated is uniformly distributed because f is a permutation. Moreover, at most $q_H + q_{sig}$ values of ω can appear during the reduction. Therefore the probability that the reduction aborts when answering an H-oracle query or a signature query is at most $(q_H + q_{sig}) \cdot 2^{-k_1}$, which gives:

$$\Pr[\mathsf{SBad}] \leq (q_H + q_{sig})^2 \cdot 2^{-k_1}$$

We denote by Bad the event:

$$\mathsf{Bad} = \mathsf{AskG} \vee \mathsf{AskH} \vee \mathsf{DBad} \vee \mathsf{SBad}$$

Let S denote the event: "the adversary outputs the correct value for b, *i.e.* $b = b'$". Conditioned on $\neg\mathsf{Bad}$, our simulation of oracles G, H, D and of the signing oracle are independent of b, and therefore the adversary's view is independent of b. This gives:

$$\Pr[\mathsf{S}|\neg\mathsf{Bad}] = \frac{1}{2} \tag{16}$$

Moreover, conditioned on $\neg\mathsf{Bad}$, the adversary's view is the same as when interacting with (perfect) random oracles, decryption oracle and signing oracle, which gives:

$$\Pr[\mathsf{S} \wedge \neg\mathsf{Bad}] \geq \frac{1}{2} + \frac{\varepsilon}{2} - \Pr[\mathsf{Bad}] \tag{17}$$

which yields as in Lemma 1:

$$\Pr[\mathsf{Bad}] \geq \varepsilon \tag{18}$$

and eventually:

$$\Pr[\mathsf{AskG}] \geq \varepsilon - (q_H + q_{sig}) \cdot 2^{-k_0} - q_D \cdot 2^{-k_1} - (q_H + q_{sig})^2 \cdot 2^{-k_1}$$

B Proof of Theorem 3

From $\mathcal{A}$ we construct an algorithm $\mathcal{B}$, which receives as input c and outputs η such that $c = f(\eta)$.

Top-level description of the reduction $\mathcal{B}$:

1. $\mathcal{B}$ is given a function f and $c = f(\eta)$, for a random integer η.
2. $\mathcal{B}$ selects uniformly at random an integer $j \in [1, q_H]$.
3. $\mathcal{B}$ runs $\mathcal{A}$ with f. It simulates the decryption oracle, the signing oracle and random oracles H and G as described below. $\mathcal{B}$ maintains a counter i for the i-th query $M_i \| r_i$ to H. The oracles H and G are simulated in such a way that if $i = j$ then $\mu(M_i \| r_i) = c$.
4. $\mathcal{B}$ receives from $\mathcal{A}$ a forgery σ. Letting M and r be the corresponding message and random, if $(M, r) = (M_j, r_j)$ then $f(\sigma) = \mu(M_j \| r_j) = c$ and $\mathcal{B}$ outputs σ.

Simulation of the oracles G, H, D and signing oracle:

When receiving the i-th query $M_i \| r_i$ to H, we distinguish two cases: if $i \neq j$, we generate a random $x_i \in \{0,1\}^k$ and compute $y_i = f(x_i)$. If $i = j$, we let $y_i = c$. In both cases we denote $y_i = \omega_i \| s_i$. If ω_i never appeared before, we let $G(\omega_i) = s_i \oplus (M_i \| r_i)$ and return ω_i, otherwise we abort.

When receiving a query ω for G, if $G(\omega)$ has already been defined, we return $G(\omega)$, otherwise we return a random $(k - k_1)$-bit integer.

When we receive a signature query for M, we generate a random k_0-bit integer r. If $M \| r$ was queried to H before, we have $M \| r = M_i \| r_i$ for some i. If $i \neq j$, we have:

$$H(M_i \| r_i) = \omega_i, \quad G(\omega_i) = s_i \oplus (M_i \| r_i) \quad \text{and} \quad y_i = \omega_i \| s_i = f(x_i)$$

so we return the corresponding signature x_i, otherwise we abort. If $M \| r$ was never queried before, we simulate an H-query for $M \| r$ as previously: we generate a random $x \in \{0,1\}^k$ and compute $y = f(x)$. We denote $y = \omega \| s$. If ω never appeared before, we let $H(M \| r) = \omega$ and $G(\omega) = s \oplus (M \| r)$ and return the signature x, otherwise we abort.

The simulation of the decryption oracle is identical to that of Lemma 1.

Analysis:

Let σ be the forgery sent by the adversary. If ω was not queried to G, we simulate a query to G as previously. Let $\omega \| s = f(\sigma)$ and $M \| r = G(\omega) \oplus s$. If $M \| r$ was never queried to H, then $H(M \| r)$ is undefined because there was no signature query for M;

the probability that $H(M\|r) = \omega$ is then 2^{-k_1}. Otherwise, let $(M, r) = (M_i, r_i)$ be the corresponding query to H. If $i = j$, then $\mu(M_j, r_j) = c = f(\sigma)$ and $\mathcal{B}$ succeeds in inverting f.

Conditioned on $i = j$, our simulation of H and the signing oracle are perfect, unless some ω appears twice, which happens with probability less than $(q_H + q_{sig})^2 \cdot 2^{-k_1}$. As in lemma 1, our simulation of D fails with probability less than $q_D \cdot 2^{-k_1}$. Consequently, the reduction $\mathcal{B}$ succeeds with probability greater than:

$$\frac{1}{q_H} \cdot \left(\varepsilon - 2^{-k_1} - (q_H + q_{sig})^2 \cdot 2^{-k_1} - q_D \cdot 2^{-k_1}\right)$$

which gives (10).

C Proof of Lemma 3

Let $\boldsymbol{b_1}, \ldots, \boldsymbol{b_d} \in \mathbb{Z}^n$ be linearly independent vectors. A *lattice* L spanned by the vectors $(\boldsymbol{b_1}, \ldots, \boldsymbol{b_d})$ is the set of all integer linear combinations of $\boldsymbol{b_1}, \ldots, \boldsymbol{b_d}$. The integer d is called the *rank* of the lattice. We say that the lattice is of full rank if $n = d$. We denote by $\|L\|_\infty$ the infinite norm of the shortest non-zero vector of L.

Given $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{n-1}) \in (\mathbb{Z}_N)^{n-1}$, consider the set:

$$L(\boldsymbol{\alpha}) = \{\boldsymbol{s} = (s_0, \ldots, s_{n-1}) \in \mathbb{Z}^n | s_i - \alpha_i \cdot s_0 = 0 \bmod N \text{ for all } 1 \leq i \leq n-1\}$$

The set L is a full rank lattice spanned by the n vectors:

$$(1, \alpha_1, \ldots, \alpha_{n-1}), (0, N, 0, \ldots, 0), \cdots, (0, \ldots, 0, N) \tag{19}$$

The proof of lemma 3 is based on the following three *lemmata*:

Lemma 5. *The probability over* $\boldsymbol{\alpha} \in \mathbb{Z}_N^{n-1}$ *that* $\|L(\boldsymbol{\alpha})\|_\infty < C$ *is less than*

$$\frac{(3C)^n}{N^{n-1}}$$

Proof. To each $L(\boldsymbol{\alpha})$ such that $\|L(\boldsymbol{\alpha})\|_\infty < C$ we can associate a shortest vector $\boldsymbol{b}(\boldsymbol{\alpha})$ such that $\|\boldsymbol{b}(\boldsymbol{\alpha})\|_\infty < C$. There are at most $(2C+1)^n$ such vectors.

Let b_0 be the first component of $\boldsymbol{b}(\boldsymbol{\alpha})$. If $b_0 = 0 \bmod N$, then all the components of $\boldsymbol{b}(\boldsymbol{\alpha})$ are equal to zero modulo N. This gives $\|\boldsymbol{b}(\boldsymbol{\alpha})\|_\infty \geq N$, and the first vector in (19) is shorter for $\|\|_\infty$ than $\boldsymbol{b}(\boldsymbol{\alpha})$. Therefore $b_0 \neq 0 \bmod N$.

If b_0 is invertible modulo N, this uniquely determines $\boldsymbol{\alpha}$. Otherwise, let p and q be the prime factors of N. If $b_0 = 0 \bmod p$, then $b_0 \neq 0 \bmod q$ and this uniquely determines $\boldsymbol{\alpha}$ modulo q, so there are at most p^{n-1} possible values for $\boldsymbol{\alpha}$. Moreover, all the components of $\boldsymbol{b}(\boldsymbol{\alpha})$ are equal to 0 modulo p, and for any C the number of such vectors $\boldsymbol{b}(\boldsymbol{\alpha})$ is at most:

$$\left(2 \cdot \left\lfloor \frac{C}{p} \right\rfloor + 1\right)^n - 1 \leq \left(\frac{3C}{p}\right)^n$$

which corresponds to at most:

$$\left(\frac{3C}{p}\right)^n \cdot p^{n-1} = \frac{(3C)^n}{p}$$

possible values for $\boldsymbol{\alpha}$. The same holds if $b_0 = 0 \bmod q$. Therefore there are at most:

$$(2C+1)^n + (3C)^n \cdot \left(\frac{1}{p} + \frac{1}{q}\right) \leq (3C)^n$$

vectors $\boldsymbol{\alpha}$ such that $\|L(\boldsymbol{\alpha})\|_\infty < C$. □

Lemma 6. *If $\|L(\boldsymbol{\alpha})\|_\infty \geq 2 \cdot B$, then the solution $\boldsymbol{s}$ of the system $\mathcal{S}$ with $\|\boldsymbol{s}\|_\infty < B$ is unique and is equal to $\boldsymbol{T} - \boldsymbol{P}$, where $\boldsymbol{T} = (0, c_1, \ldots, c_{n-1})$ and $\boldsymbol{P}$ is the closest vector to $\boldsymbol{T}$ for $\|\|_\infty$.*

Proof. Let $\boldsymbol{s'}$ be another solution of $\mathcal{S}$ with $\|\boldsymbol{s'}\|_\infty < B$. Then $\boldsymbol{s} - \boldsymbol{s'} \in L(\boldsymbol{\alpha})$ and $\|\boldsymbol{s} - \boldsymbol{s'}\|_\infty < 2 \cdot B$ which gives $\boldsymbol{s} = \boldsymbol{s'}$ since $\|L(\boldsymbol{\alpha})\|_\infty \geq 2 \cdot B$

Let $\boldsymbol{s'} = \boldsymbol{T} - \boldsymbol{P}$ where $\boldsymbol{P} \in L(\boldsymbol{\alpha})$ is a closest vector to $\boldsymbol{T}$ for $\|\|_\infty$. Since $\boldsymbol{T} - \boldsymbol{s}$ is a vector of $L(\boldsymbol{\alpha})$, we have:

$$\|\boldsymbol{s'}\|_\infty = \|\boldsymbol{T} - \boldsymbol{P}\|_\infty \leq \|\boldsymbol{T} - (\boldsymbol{T} - \boldsymbol{s})\|_\infty = \|\boldsymbol{s}\|_\infty$$

$$\|\boldsymbol{s'} - \boldsymbol{s}\|_\infty \leq \|\boldsymbol{s'}\|_\infty + \|\boldsymbol{s}\|_\infty \leq 2\|\boldsymbol{s}\|_\infty < 2 \cdot B$$

and so $\boldsymbol{s'} = \boldsymbol{s}$. □

Lemma 7. *Let $(\boldsymbol{b_1}, \ldots, \boldsymbol{b_n})$ be a basis of a lattice $L \subset \mathbb{Z}^n$ such that $\|L\|_\infty \geq B \cdot (1 + \sqrt{n} \cdot 2^{n/2})$ and $\boldsymbol{T}$ a vector which distance to L for $\|\|_\infty$ is strictly less than B. There exists a polynomial-time algorithm taking as input $(\boldsymbol{b_1}, \ldots, \boldsymbol{b_n})$ and $\boldsymbol{T}$ and outputting a closest vector $\boldsymbol{P} \in L$ to $\boldsymbol{T}$ for $\|\|_\infty$.*

Proof. The proof is based on the following theorem:

Theorem 7 (Babai [1]). *There exists a polynomial time algorithm which, given a basis $(\boldsymbol{b_1}, \ldots, \boldsymbol{b_n})$ of a lattice $L \subset \mathbb{Z}^n$, approximates the closest vector problem for the Euclidean norm to a factor $2^{n/2}$.*

Let $\boldsymbol{P'}$ be the vector obtained by running Babai's algorithm on $(\boldsymbol{b_1}, \ldots, \boldsymbol{b_n})$ and $\boldsymbol{T}$. Let $\boldsymbol{P}$ a closest vector to $\boldsymbol{T}$ for $\|\|_\infty$. We show that $\boldsymbol{P'} = \boldsymbol{P}$.

Letting $\boldsymbol{P''}$ be a closest vector to $\boldsymbol{T}$ for the Euclidean norm, we have:

$$\|\boldsymbol{T} - \boldsymbol{P'}\| \leq 2^{n/2}\|\boldsymbol{T} - \boldsymbol{P''}\|$$

Moreover, since $\boldsymbol{P''}$ is a closest vector to $\boldsymbol{T}$ for $\|\|$, we have:

$$\|\boldsymbol{T} - \boldsymbol{P''}\| \leq \|\boldsymbol{T} - \boldsymbol{P}\|$$

The distance of $\boldsymbol{T}$ to L for $\|\|_\infty$ is strictly less than B, therefore:

$$\|\boldsymbol{T} - \boldsymbol{P}\|_\infty < B$$

This gives:

$$\|\boldsymbol{T}-\boldsymbol{P'}\|_\infty \leq \|\boldsymbol{T}-\boldsymbol{P'}\| \leq 2^{n/2}\|\boldsymbol{T}-\boldsymbol{P}\| \leq \sqrt{n}\cdot 2^{n/2}\|\boldsymbol{T}-\boldsymbol{P}\|_\infty < B\sqrt{n}\cdot 2^{n/2}$$

and eventually

$$\|\boldsymbol{P}-\boldsymbol{P'}\|_\infty \leq \|\boldsymbol{P}-\boldsymbol{T}\|_\infty + \|\boldsymbol{T}-\boldsymbol{P'}\|_\infty < B(1+\sqrt{n}\cdot 2^{n/2})$$

and so $\boldsymbol{P}=\boldsymbol{P'}$. □

Resuming the proof of lemma 3, we take $B = 2^{k_2}$ and $C = 2^{k_2}(1+\sqrt{n}\cdot 2^{n/2})$. We consider the lattices $L(\boldsymbol{\alpha})$ such that $\|L(\boldsymbol{\alpha})\|_\infty \geq C$. From lemma 5, and using:

$$3C \leq 2^{k_2+n+2}$$

the proportion of lattices $L(\boldsymbol{\alpha})$ such that $\|L(\boldsymbol{\alpha})\|_\infty < C$ is smaller than:

$$\frac{2^{n\cdot(k_2+n+2)}}{N^{n-1}}$$

From lemma 6 the solution $\boldsymbol{s}$ of the system $\mathcal{S}$ with $\|\boldsymbol{s}\|_\infty < 2^{k_2}$ is unique and equal to $\boldsymbol{T}-\boldsymbol{P}$, where $\boldsymbol{T}=(0,c_1,\ldots,c_{n-1})$ and $\boldsymbol{P}$ is the closest vector to $\boldsymbol{T}$ for $\|\|_\infty$. From lemma 7, and using the basis (19) for $L(\boldsymbol{\alpha})$, we can compute $\boldsymbol{P}$ in time polynomial in n and in the size of N.

D Proof of Inequality (14)

We assume that:

$$k_1 \geq 64 \quad \text{and} \quad k \leq 2^{-6}\cdot (k_1)^2 \tag{20}$$

We have:

$$\begin{aligned}\frac{2^{n\cdot(k_2+n+2)}}{N^{n-1}} &\leq 2^{n\cdot(k-k_1+n+2)-(n-1)\cdot(k-1)}\\ &\leq 2^{n\cdot(-k_1+n+3)+k-1}\end{aligned}$$

Letting $f(x) = x\cdot(-k_1+x+3)+k-1$, we have $f'(x) = -k_1+2\cdot x+3$. For $0 \leq x \leq 5k/(4k_1)+1$ and using (20), we obtain $f'(x) \leq 0$. We take:

$$n = \left\lceil \frac{5k}{4k_1} \right\rceil$$

f is then a decreasing function for $0 \leq x \leq n$, therefore:

$$f\left(\frac{5k}{4k_1}\right) \geq f(n)$$

which yields using (20):

$$f(n) \leq -k/8$$

from which we obtain inequality (14).

E Proof of Theorem 6

The proof is similar to the security proof of PSS in [4]. The only difference is that we simulate a decryption oracle as in theorem 3. This adds an error probability of $q_D\cdot 2^{-k_1}$.

Security Proof for Partial-Domain Hash Signature Schemes

Proceedings of Crypto 2002

Jean-Sébastien Coron

Gemplus Card International
34 rue Guynemer
Issy-les-Moulineaux, F-92447, France
jean-sebastien.coron@gemplus.com

Abstract. We study the security of partial-domain hash signature schemes, in which the output size of the hash function is only a fraction of the modulus size. We show that for $e = 2$ (Rabin), partial-domain hash signature schemes are provably secure in the random oracle model, if the output size of the hash function is larger than 2/3 of the modulus size. This provides a security proof for a variant of the signature standards ISO 9796-2 and PKCS#1 v1.5, in which a larger digest size is used.

Key-words: Signature Schemes, Provable Security, Random Oracle Model.

1 Introduction

A common practice for signing with RSA or Rabin consists in first hashing the message m, then padding the hash value with some predetermined or message-dependent block, and eventually raising the result $\mu(m)$ to the private exponent d. This is commonly referred to as the "hash-and-sign" paradigm:

$$s = \mu(m)^d \bmod N$$

For digital signature schemes, the strongest security notion was defined by Goldwasser, Micali and Rivest in [8], as *existential unforgeability under an adaptive chosen message attack*. This notion captures the property that an attacker cannot produce a valid signature, even after obtaining the signature of (polynomially many) messages of his choice.

The random oracle model, introduced by Bellare and Rogaway in [2], is a theoretical framework allowing to prove the security of hash-and-sign signature schemes. In this model, the hash function is seen as an oracle which outputs a random value for each new query. Bellare and Rogaway defined in [3] the Full Domain Hash (FDH) signature scheme, in which the output size of the hash function is the same as the modulus size. FDH is provably secure in the random oracle model assuming that inverting RSA is hard. Actually, a security proof in the random oracle model does not necessarily imply that the scheme is secure in the real world (see [4]). Nevertheless, it seems to be a good engineering principle to design a scheme so that it is provably secure in the random oracle model. Many encryption and signature schemes were proven to be secure in the random oracle model

Other hash-and-sign signature schemes include the widely used signature standards PKCS#1 v1.5 and ISO 9796-2. In these standards, the digest size is only a fraction of the modulus size. As opposed to FDH, no security proof is known for those standards.

Moreover, it was shown in [5] that ISO 9796-2 was insecure if the size of the hash function was too small, and the standard was subsequently revised.

In this paper, we study the security of partial-domain hash signature schemes, in which the hash size is only a fraction of the modulus size. We show that for $e = 2$, partial-domain hash signature schemes are provably secure in the random oracle model, assuming that factoring is hard, if the size of the hash function is larger than 2/3 of the modulus size. The proof is based on a modification of Vallée's generator of small random squares [16]. This provides a security proof for a variant of PKCS#1 v1.5 and ISO 9796-2 signatures, in which the digest size is larger than 2/3 of the size of the modulus.

2 Definitions

In this section we briefly present some notations and definitions used throughout the paper. We start by recalling the definition of a signature scheme.

Definition 1 (Signature Scheme). *A signature scheme* $(\texttt{Gen}, \texttt{Sign}, \texttt{Verify})$ *is defined as follows:*

- The key generation algorithm `Gen` *is a probabilistic algorithm which given* 1^k*, outputs a pair of matching public and private keys,* (pk, sk)*.*

- The signing algorithm `Sign` *takes the message* M *to be signed, the private key* sk*, and returns a signature* $x = \texttt{Sign}_{sk}(M)$*. The signing algorithm may be probabilistic.*

- The verification algorithm `Verify` *takes a message* M*, a candidate signature* x' *and* pk*. It returns a bit* $\texttt{Verify}_{pk}(M, x')$*, equal to one if the signature is accepted, and zero otherwise. We require that if* $x \leftarrow \texttt{Sign}_{sk}(M)$*, then* $\texttt{Verify}_{pk}(M, x) = 1$*.*

In the previously introduced *existential unforgeability under an adaptive chosen message attack* scenario, the forger can dynamically obtain signatures of messages of his choice and attempt to output a valid forgery. A *valid forgery* is a message/signature pair (M, x) such that $\texttt{Verify}_{pk}(M, x) = 1$ whereas the signature of M was never requested by the forger. Moreover, in the random oracle model, the attacker cannot evaluate the hash function by himself; instead, he queries an oracle which outputs a random value for each new query.

RSA [14] is undoubtedly the most widely used cryptosystem today:

Definition 2 (RSA). *The RSA cryptosystem is a family of trapdoor permutations, specified by:*

- The RSA generator $\mathcal{RSA}$*, which on input* 1^k*, randomly selects two distinct* $k/2$*-bit primes* p *and* q *and computes the modulus* $N = p \cdot q$*. It picks an encryption exponent* $e \in \mathbb{Z}^*_{\phi(N)}$ *and computes the corresponding decryption exponent* d *such that* $e \cdot d = 1 \bmod \phi(N)$*. The generator returns* (N, e, d)*.*

- The encryption function $f : \mathbb{Z}^*_N \rightarrow \mathbb{Z}^*_N$ *defined by* $f(x) = x^e \bmod N$*.*

- The decryption function $f^{-1} : \mathbb{Z}^*_N \rightarrow \mathbb{Z}^*_N$ *defined by* $f^{-1}(y) = y^d \bmod N$*.*

An *inverting algorithm* $\mathcal{I}$ for RSA gets as input (N, e, y) and tries to find $y^d \bmod N$. Its success probability is the probability to output $y^d \bmod N$ when (N, e, d) are obtained by running $\mathcal{RSA}(1^k)$ and y is set to $x^e \bmod N$ for some x chosen at random in $\mathbb{Z}^*_N$.

The Full-Domain-Hash scheme (FDH) [3] was the first practical and provably secure signature scheme based on RSA. It is defined as follows: the key generation algorithm, on input 1^k, runs $\mathcal{RSA}(1^k)$ to obtain (N, e, d). It outputs (pk, sk), where the public key pk is (N, e) and the private key sk is (N, d). The signing and verifying algorithms use a hash function $H : \{0,1\}^* \to \mathbb{Z}_N^*$ which maps bit strings of arbitrary length to the set of invertible integers modulo N.

$\mathtt{SignFDH}_{N,d}(M)$
  $y \leftarrow H(M)$
  return $y^d \bmod N$

$\mathtt{VerifyFDH}_{N,e}(M, x)$
  $y \leftarrow x^e \bmod N$
  if $y = H(M)$ then return 1 else return 0.

The following theorem [6] proves the security of FDH in the random oracle model, assuming that inverting RSA is hard. It provides a better security bound than [3].

Theorem 1. *Assume that there is no algorithm which inverts RSA with probability greater than ε within time t. Then the success probability of a FDH forger making at most q_{hash} hash queries and q_{sig} signature queries within running time t' is less than ε', where*

$$\varepsilon' = 4 \cdot q_{sig} \cdot \varepsilon$$
$$t' = t - (q_{hash} + q_{sig} + 1) \cdot \mathcal{O}(k^3)$$

We say that a hash-and-sign signature scheme is a *partial-domain hash signature scheme* if the encoding function $\mu(m)$ can be written as:

$$\mu(m) = \gamma \cdot H(m) + f(m) \tag{1}$$

where γ is a constant, H a hash function and f some function of m. A typical example of a partial-domain hash signature scheme is the ISO 9796-2 standard with full message recovery [11]:

$$\mu(m) = \mathtt{4A_{16}} \| m \| H(m) \| \mathtt{BC_{16}}$$

The main result of this paper is to show that for $e = 2$, partial-domain hash signature schemes are provably secure, if the hash size is larger than 2/3 of the modulus size. In the following, we recall the Rabin-Williams signature scheme [12]. It uses a padding function $\mu(m)$ such that for all m, $\mu(m) = 6 \bmod 16$.

- Key generation: on input 1^k, generate two $k/2$-bit primes p and q such that $p = 3 \bmod 8$ and $q = 7 \bmod 8$. The public key is $N = p \cdot q$ and the private key is $d = (N - p - q + 5)/8$.

- Signature generation: compute the Jacobi symbol

$$J = \left(\frac{\mu(m)}{N}\right)$$

The signature of m is $s = \min(\sigma, N - \sigma)$, where:

$$\sigma = \begin{cases} \mu(m)^d \bmod N & \text{if } J = 1 \\ (\mu(m)/2)^d \bmod N & \text{otherwise} \end{cases}$$

- Signature verification: compute $\omega = s^2 \bmod N$ and check that:

$$\mu(m) \stackrel{?}{=} \begin{cases} \omega & \text{if } \omega = 6 \bmod 8 \\ 2 \cdot \omega & \text{if } \omega = 3 \bmod 8 \\ N - \omega & \text{if } \omega = 7 \bmod 8 \\ 2 \cdot (N - \omega) & \text{if } \omega = 2 \bmod 8 \end{cases}$$

3 Security of Partial-domain Hash Signature Schemes

To prove the security of a signature scheme against chosen message attacks, one must be able to answer the signature queries of the attacker. In FDH's security proof, when answering a hash query, one generates a random $r \in \mathbb{Z}_N$ and answers $H(m) = r^e \bmod N$ so that the signature r of m is known. Similarly, for partial-domain hash signature schemes, we should be able to generate a random r such that:

$$\mu(m) = \gamma \cdot H(m) + f(m) = r^e \bmod N$$

with $H(m)$ being uniformly distributed in the output space of the hash function. For example, if we take $\mu(m) = H(m)$ where $0 \leq H(m) \leq N^\beta$ and $\beta < 1$, one should be able to generate a random r such that $r^e \bmod N$ is uniformly distributed between 0 and N^β.

Up to our knowledge, no such algorithm is known for $e \geq 3$. For $e = 2$, Vallée constructed in [16] a random generator where the size of $r^2 \bmod N$ is less than $2/3$ of the size of the modulus. [16] used this generator to obtain proven complexity bounds for the quadratic sieve factoring algorithm. Vallée's generator has a quasi-uniform distribution; a distribution is said to be *quasi-uniform* if there is a constant ℓ such that for all x, the probability to generate x lies between $1/\ell$ and ℓ times the probability to generate x under the uniform distribution. However, quasi-uniformity is not sufficient here, as we must simulate a random oracle and therefore our simulation should be indistinguishable from the uniform distribution.

Our contribution is to modify Vallée's generator in order to generate random squares in any interval of size $N^{2/3+\varepsilon}$, with a distribution which is statistically indistinguishable from the uniform distribution. From this generator we will derive a security proof for partial-domain hash signatures, in which the digest size is at least $2/3$ of the modulus size.

Remark: for Paillier's trapdoor permutation [13] with parameter $g = 1 + N$, it is easy to show that half-domain hash is provably secure in the random oracle model, assuming that inverting RSA with $e = N$ is hard.

4 Generating Random Squares in a Given Interval

4.1 Notations

We identify $\mathbb{Z}_N$, the ring of integers modulo N with the set of integers between 0 and $N - 1$. We denote by $\mathbb{Z}_N^+$ the set of integers between 0 and $(N - 1)/2$. We denote by Q the squaring operation over $\mathbb{Z}_N$:

$$Q(x) = x^2 \bmod N$$

Given positive integers a and h such that $a + h < N$, let B be the set:

$$B = \{x \in \mathbb{Z}_N^+ \mid a \leq Q(x) \leq a + h\}$$

Our goal is to generate integers $x \in B$ with a distribution statistically indistinguishable from the uniform distribution. The *statistical distance* between two distributions X and Y is defined as the function:

$$\delta = \frac{1}{2} \sum_{\alpha} |\Pr[X = \alpha] - \Pr[Y = \alpha]|$$

We say that two ensembles $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$ are *statistically indistinguishable* if their statistical distance δ_n is a negligible function of n.

4.2 Description of B

In this section, we recall Vallée's description of the set B. We denote by b the cardinality of B. The following lemma, which proof can be derived from equation (6) in [16], shows that b is close to $h/2$.

Lemma 1. *Let N be a ℓ-bit RSA modulus. We have for $\ell \geq 64$:*

$$\left| b - \frac{h}{2} \right| \leq 4 \cdot \ell \cdot 2^{\ell/2}$$

In the following, we assume that the bit size of N is greater than 64. As in [16], we introduce Farey sequences [9]:

Definition 3 (Farey sequence). *The Farey sequence $\mathcal{F}_k$ of order k is the ascending sequence of irreducible fractions between 0 and 1 whose denominators do not exceed k. Thus p/q belongs to $\mathcal{F}_k$ if $0 \leq p \leq q \leq k$ and $\gcd(p, q) = 1$.*

The characteristic property of Farey sequences is expressed by the following theorem [9]:

Theorem 2. *If p/q and p'/q' are two successive terms of $\mathcal{F}_k$, then $q \cdot p' - p \cdot q' = 1$*

Given $p/q \in \mathcal{F}_k$, we define the *Farey interval* $I(p, q)$ as the interval of center $pN/(2q)$ and radius $N/(2kq)$. Given the terms p'/q' and p''/q'' of $\mathcal{F}_k$ which precede and follow p/q, we let $J(p, q)$ be the interval:

$$J(p, q) = \left[\frac{N(p + p')}{2(q + q')}, \frac{N(p + p'')}{2(q + q'')} \right]$$

If $p/q = 0/1$, then p/q has no predecessor and we take $p'/q' = 0/1$. Similarly, if $p/q = 1/1$, we take $p''/q'' = 1/1$. The set of intervals $J(p, q)$ forms a partition of $\mathbb{Z}_N^+$. The following lemma [16] shows that intervals $I(p, q)$ and $J(p, q)$ are closely related.

Lemma 2. *$I(p, q)$ contains $J(p, q)$ and its length is at most twice the length of $J(p, q)$.*

Given $p/q \in \mathcal{F}_k$ with $p/q \neq 0/1$, let x_0 be the integer nearest to the rational $pN/2q$:

$$x_0 - \frac{pN}{2q} = u_0 \quad \text{with } |u_0| \leq \frac{1}{2}$$

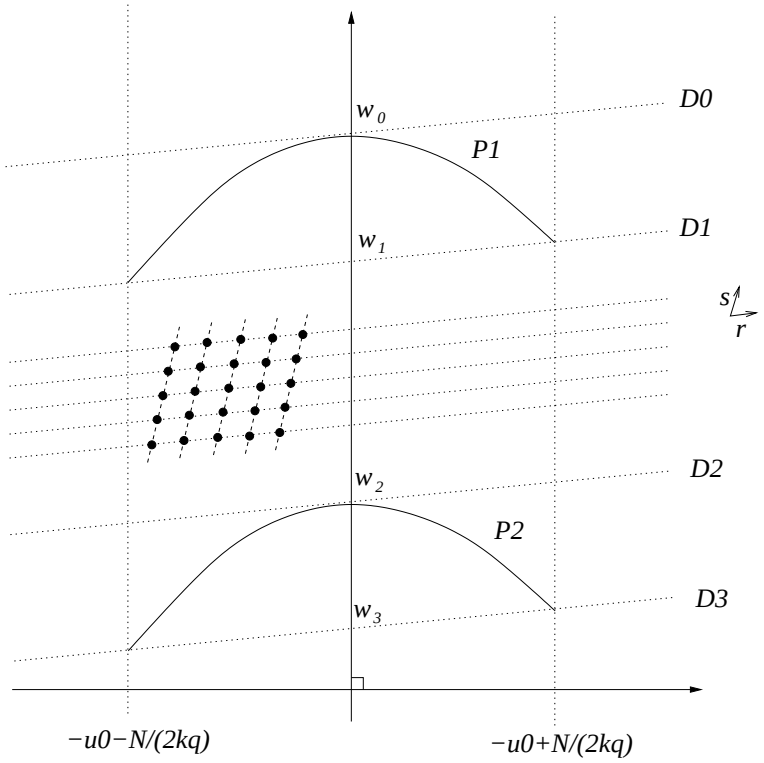

Fig. 1. The intersection between the lattice $L(x_0)$ and the domain between the two parabolas $\mathcal{P}_1$ and $\mathcal{P}_2$

Let $L(x_0)$ be the lattice spanned by the two vectors $(1, 2x_0)$ and $(0, N)$. Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be the two parabolas of equations:

$$\mathcal{P}_1 : \omega + u^2 + x_0^2 = a + h \quad \text{and} \quad \mathcal{P}_2 : \omega + u^2 + x_0^2 = a$$

Let P be the domain of lattice points comprised between the two parabolas:

$$P = \{(u, \omega) \in L(x_0) \mid a \leq \omega + u^2 + x_0^2 \leq a + h\}$$

The following lemma, which proof is straightforward, shows that the elements of B arise from the intersection of the lattice $L(x_0)$ and the domain comprised between the two parabolas (see figure 1).

Lemma 3. *$x = x_0 + u$ belongs to B iff there exists a unique ω such that the point (u, ω) belongs to P.*

We let $B(p, q)$ be the set of integers in $B \cap J(p, q)$. From Lemma 3 the integers in $B(p, q)$ arise from the domain of lattice points:

$$P(p, q) = \{(u, \omega) \in P \mid x_0 + u \in J(p, q)\}$$

From Lemma 2, the set $P(p, q)$ is included inside the set of lattice points:

$$Q(p, q) = \{(u, \omega) \in P \mid x_0 + u \in I(p, q)\}$$

whose abcissae u are comprised between $-u_0 - N/(2kq)$ and $-u_0 + N/(2kq)$. In the following, we describe the domain $Q(p,q)$, using the following two short vectors of $L(x_0)$ (see figure 1):

$$\boldsymbol{r} = q(1, 2x_0) - p(0, N) = (q, 2qu_0) \tag{2}$$

$$\boldsymbol{s} = q'(1, 2x_0) - p'(0, N) = (q', 2q'u_0 + N/q) \tag{3}$$

where p'/q' is the term of $\mathcal{F}_k$ which precedes p/q.

We consider the lines of the lattice parallel to vector $\boldsymbol{r}$ which intersect the domain $Q(p,q)$. These lines have a slope equal to $2u_0$. The first extremal position of these lines is the tangent D_0 to the first parabola:

$$D_0 : \ \omega - (-u_0^2 - x_0^2 + a + h) = 2u_0(u + u_0)$$

The second extremal position joins the two points of the second parabola with abscissae $-u_0 - N/(2kq)$ and $-u_0 + N/(2kq)$. This line D_3 has also a slope equal to $2u_0$ and satisfies the equation:

$$\omega + (u_0 + \frac{N}{2kq})^2 - a + x_0^2 = 2u_0(u + u_0 + \frac{N}{2kq})$$

The two lines intersect the vertical axis at the respective points:

$$\omega_0 = a - x_0^2 + u_0^2 + h \ \text{ and } \ \omega_3 = a - x_0^2 + u_0^2 - \frac{N^2}{4k^2q^2}$$

All the lines parallel to $\boldsymbol{r}$ that intersect $P(p,q)$ are the ones that intersect the segment $[\omega_3, \omega_0]$ on the vertical axis. We denote by $D(\nu)$ a line parallel to $\boldsymbol{r}$ which intersects the vertical axis at ordinate equal to $\omega_0 - \nu N/q$. The line D_0 is the line $D(\nu_0 = 0)$, whereas the line D_3 is the line $D(\nu_3)$ such that:

$$\nu_3 = \frac{hq}{N} + \frac{N}{4k^2q} \tag{4}$$

Eventually, we denote by $D_1 = D(\nu_1)$ the line which joins the two points of the first parabola with abcissae $-u_0 - N/(2kq)$ and $-u_0 + N/(2kq)$, and by $D_2 = D(\nu_2)$ the tangent to the second parabola, with a slope equal to $2u_0$. We have:

$$\nu_1 = \frac{N}{4k^2q} \ \text{ and } \ \nu_2 = \frac{hq}{N} \tag{5}$$

A real ν is called an index if $D(\nu)$ is a line of $L(x_0)$. The difference between two consecutive indices is equal to one.

4.3 Our New Generator

In this section, we describe our new generator of integers in B. The difference with Vallée's generator is that we use different parameters for k and h, and we do not generate all the integers in B; instead we avoid a negligible subset of B.

First, we describe a generator $\mathcal{G}(p,q)$ of integers in $B(p,q)$, and we show that its distribution is statistically indistinguishable from the uniform distribution. We assume

that $N \leq 2 \cdot k \cdot q \cdot \sqrt{h}$, which gives $\nu_1 \leq \nu_2$. Therefore the line D_1 is above the line D_2 (see figure 1). We restrict ourselves to the integers in $B(p,q)$ such that the corresponding points $(u,\omega) \in P(p,q)$ lie on $D(\nu)$ with $\nu_1 \leq \nu \leq \nu_2$. These points are the points on $D(\nu)$ whose abscissae u are such that $x_0 + u \in J(p,q)$.

Generator $\mathcal{G}(p,q)$ of integers in $B(p,q)$:
1. Generate a random index ν uniformly distributed between ν_1 and ν_2.
2. Generate a point $(u,\omega) \in P(p,q)$ on $D(\nu)$ such that $x_0 + u \in J(p,q)$, with the uniform distribution.
3. Output $x_0 + u$.

The following lemma shows that under some conditions on k, h and q, the cardinality $b(p,q)$ of $B(p,q)$ is close to $h \cdot j(p,q)/N$, where $j(p,q)$ is the number of integers in the interval $J(p,q)$. Moreover, under the same conditions, the distribution induced by $\mathcal{G}(p,q)$ is statistically indistinguishable from the uniform distribution in $B(p,q)$. The proof is given in appendix A.

Lemma 4. *Let $\alpha > 0$ and $k = N^{\frac{1}{3}-\alpha}$. Assume that $k \geq 6$, $N^{\alpha} \geq 3$ and $N^{\frac{2}{3}+13\cdot\alpha} \leq h < N$. Then for all $p/q \in \mathcal{F}_k$ such that $N^{1/3-4\alpha} \leq q \leq k$, we have:*

$$\left| b(p,q) - \frac{h \cdot j(p,q)}{N} \right| \leq \frac{4h \cdot j(p,q)}{N} N^{-3\alpha} \tag{6}$$

Moreover, $\mathcal{G}(p,q)$ generates elements in $B(p,q)$ with a distribution whose distance δ_G from the uniform distribution is at most $7 \cdot N^{-3\alpha}$.

Now we construct a generator $\mathcal{V}$ of $p/q \in \mathcal{F}_k$ such that the probability to generate p/q is close to $b(p,q)/b$. It only generates $p/q \in \mathcal{F}_k$ such that $q \geq N^{1/3-4\alpha}$, so that from the previous lemma, $b(p,q)$ is nearly proportional to the number of integers in $J(p,q)$, and the distribution induced by $\mathcal{G}(p,q)$ is close to the uniform distribution.

Generator $\mathcal{V}$ of $p/q \in \mathcal{F}_k$
1. Generate a random integer $x \in \mathbb{Z}_N^+$ with the uniform distribution.
2. Determine which interval $J(p,q)$ contains x.
3. If $q \geq N^{1/3-4\alpha}$ then output $p/q \in \mathcal{F}_k$, otherwise output $\perp$.

Lemma 5. *Let denote by $\mathcal{D}$ the distribution induced by choosing $p/q \in \mathcal{F}_k$ with probability $b(p,q)/b$. Under the conditions of lemma 4, the statistical distance δ_V between $\mathcal{D}$ and the distribution induced by $\mathcal{V}$ is at most $9 \cdot N^{-3\alpha}$.*

Proof. See appendix B.

Eventually, our generator $\mathcal{G}$ of elements in B combines the two generators $\mathcal{V}$ and $\mathcal{G}(p,q)$:

Generator $\mathcal{G}$ of $x \in B$
1. Generate y using $\mathcal{V}$.
2. If $y = \perp$, then output $\perp$.
3. Otherwise, $y = p/q$ and generate $x \in B(p,q)$ using $\mathcal{G}(p,q)$. Output x.

The following theorem, whose proof is given in appendix C, shows that the distribution induced by $\mathcal{G}$ is statistically indistinguishable from the uniform distribution in B.

Theorem 3. *For any $\varepsilon > 0$, letting $h = N^{\frac{2}{3}+\varepsilon}$ and $\alpha = \varepsilon/13$. If $N^{\alpha} \geq 3$, then the distance δ between the distribution induced by $\mathcal{G}$ and the uniform distribution in B is at most $16 \cdot N^{-3 \cdot \varepsilon/13}$. The running time of $\mathcal{G}$ is $\mathcal{O}(\log^3 N)$.*

5 A Security Proof for Partial-domain Hash Signature Schemes

In this section, using the previous generator $\mathcal{G}$ of random squares, we show that partial-domain hash signature schemes are provably secure in the random oracle model, for $e = 2$, assuming that factoring is hard, if the size of the hash function is larger than 2/3 of the modulus size. Moreover, we restrict ourselves to small constants γ in (1), *e.g.* $\gamma = 16$ or $\gamma = 256$. This is the case for all the signature standards of the next section. We denote by k_0 the hash function's digest size. The proof is similar to the proof of theorem 1 and is given in the full version of this paper [7].

Theorem 4. *Let $\mathcal{S}$ be the Rabin-Williams partial-domain hash signature scheme with constant γ and hash size k_0 bits. Assume that there is no algorithm which factors a RSA modulus with probability greater than ε within time t. Then the success probability of a forger against $\mathcal{S}$ making at most q_{hash} hash queries and q_{sig} signature queries within time t' is upper bounded by ε', where:*

$$\varepsilon' = 8 \cdot q_{sig} \cdot \varepsilon + 32 \cdot (q_{hash} + q_{sig} + 1) \cdot k_1 \cdot \gamma \cdot 2^{-\frac{3}{13} \cdot k_1} \tag{7}$$

$$t' = t - k_1 \cdot \gamma \cdot (q_{hash} + q_{sig} + 1) \cdot \mathcal{O}(k^3) \tag{8}$$

and $k_1 = k_0 - \frac{2}{3}k$.

6 Application to Signature Standards

6.1 PKCS#1 v1.5 and SSL-3.02

The signature scheme PKCS#1 v1.5 [15] is a partial-domain hash signature scheme, with:

$$\mu(m) = \mathtt{0001}_{16} \| \mathtt{FFFF}_{16} \ldots \mathtt{FFFF}_{16} \| \mathtt{00}_{16} \| c_{\mathrm{SHA}} \| H(m)$$

where c_{SHA} is a constant and $H(m) = \mathrm{SHA}(m)$, or

$$\mu(m) = \mathtt{0001}_{16} \| \mathtt{FFFF}_{16} \ldots \mathtt{FFFF}_{16} \| \mathtt{00}_{16} \| c_{\mathrm{MD5}} \| H(m)$$

where c_{MD5} is a constant and $H(m) = \mathrm{MD5}(m)$.

The standard PKCS#1 v1.5 was not designed to work with Rabin ($e = 2$). However, one can replace the last nibble of $H(m)$ by 6 and obtain a padding scheme which is compatible with the Rabin-Williams signature scheme. The standard is then provably secure if the size of the hash-function is larger than 2/3 of the size of the modulus. This is much larger than the 128 or 160 bits which are recommended in the standard. The same analysis applies for the SSL-3.02 padding scheme [10].

6.2 ISO 9796-2 and ANSI x9.31

The ISO 9796-2 encoding scheme [11] is defined as follows:

$$\mu(m) = \mathtt{6A}_{16} \| m[1] \| H(m) \| \mathtt{BC}_{16}$$

where $m[1]$ is the leftmost part of the message, or:

$$\mu(m) = \mathtt{4A}_{16} \| m \| H(m) \| \mathtt{BC}_{16}$$

[11] describes an application of ISO 9796-2 with the Rabin-Williams signature scheme. Note that since $\mu(m) = 12 \bmod 16$ instead of $\mu(m) = 6 \bmod 16$, there is a slight change in the verification process. However, the same security bound applies: the scheme is provably secure if the size of the hash-function is larger than 2/3 of the size of the modulus. The same analysis applies for the ANSI x9.31 padding scheme [1].

7 Conclusion

We have shown that for Rabin, partial-domain hash signature schemes are provably secure in the random oracle, assuming that factoring is hard, if the size of the hash function is larger than 2/3 of the modulus size. Unfortunately, this is much larger than the size which is recommended in the standards PKCS#1 v1.5 and ISO 9796-2. An open problem is to obtain a smaller bound for the digest size, and to extend this result to RSA signatures.

Acknowledgements

I wish to thanks the anonymous referees for their helpful comments.

References

1. ANSI X9.31, *Digital signatures using reversible public-key cryptography for the financial services industry (rDSA)*, 1998.
2. M. Bellare and P. Rogaway, *Random oracles are practical : a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Commmunications Security, ACM, 1993.
3. M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*. Proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, 1996, pp. 399-416.
4. R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology, revisited*, STOC' 98, ACM, 1998.
5. J.S. Coron, D. Naccache and J.P. Stern, *On the security of RSA Padding*, Proceedings of Crypto'99, LNCS vol. 1666, Springer-Verlag, 1999, pp. 1-18.
6. J.S. Coron, *On the exact security of Full Domain Hash*, Proceedings of Crypto 2000, LNCS vol. 1880, Springer-Verlag, 2000, pp. 229-235.
7. J.S. Coron, *Security proof for partial-domain hash signature schemes*. Full version of this paper. Cryptology ePrint Archive, `http://eprint.iacr.org`.

8. S. Goldwasser, S. Micali and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of computing, 17(2):281-308, april 1988.
9. G.H. Hardy and E.M. Wright, *An introduction to the theory of numbers*, Oxford science publications, fifth edition.
10. K. Hickman, *The SSL Protocol*, December 1995. Available electronically at : `www.netscape.com/newsref/std/ssl.html`
11. ISO/IEC 9796-2, *Information technology - Security techniques - Digital signature scheme giving message recovery, Part 2 : Mechanisms using a hash-function*, 1997.
12. A.J. Menezes, P. C. van Oorschot and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC press, 1996.
13. P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, proceedings of Eurocrypt'99, LNCS 1592, pp. 223-238, 1999.
14. R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, CACM 21, 1978.
15. RSA Laboratories, PKCS #1 : *RSA cryptography specifications*, version 1.5, November 1993 and version 2.0, September 1998.
16. B. Vallée, *Generation of elements with small modular squares and provably fast integer factoring algorithms*, Mathematics of Computation, vol. 56, number 194, april 1991, pp. 823-849.

A Proof of lemma 4

From the conditions of lemma 4, we obtain:

$$\frac{hq}{N} \geq N^{9\alpha} \quad \text{and} \quad \frac{N}{k^2 q} \leq N^{6\alpha} \tag{9}$$

which gives $N \leq 2 \cdot k \cdot q \cdot \sqrt{h}$ and then $\nu_1 < \nu_2$.

Recall that $j(p,q)$ denotes the number of integers in interval $J(p,q)$. From lemma 2 the length of $J(p,q)$ is at least $N/(2kq)$ and therefore, $j(p,q) \geq N/(2kq) - 1$, which gives using $k \geq 6$:

$$\frac{j(p,q)}{q} \geq \frac{N^{3\alpha}}{3} \tag{10}$$

Let us denote by $n(\nu)$ the number of points of $P(p,q)$ on a line $D(\nu)$. The distance between the abcissae of two consecutive points of $P(p,q)$ on a line $D(\nu)$ is equal to q. Therefore, for all indices ν, we have $n(\nu) \leq \lfloor j(p,q)/q \rfloor + 1$. Moreover, for $\nu_1 \leq \nu \leq \nu_2$, $n(\nu)$ is either $\lfloor j(p,q)/q \rfloor$ or $\lfloor j(p,q)/q \rfloor + 1$. This gives the following bound for $b(p,q)$:

$$(\nu_2 - \nu_1 - 1) \cdot \left(\frac{j(p,q)}{q} - 1 \right) \leq b(p,q) \leq (\nu_3 + 1) \cdot \left(\frac{j(p,q)}{q} + 1 \right)$$

which gives using (4), (5), (9), (10) and $N^\alpha \geq 3$:

$$\left| b(p,q) - \frac{h \cdot j(p,q)}{N} \right| \leq \frac{4h \cdot j(p,q)}{N} N^{-3\alpha} \tag{11}$$

Let n' be the number of indices ν such that $\nu_1 \leq \nu \leq \nu_2$. We have $n' = \lfloor \nu_2 - \nu_1 \rfloor$ or $n' = \lfloor \nu_2 - \nu_1 \rfloor + 1$. The probability that $\mathcal{G}(p,q)$ generates an element $x \in B(p,q)$ corresponding to a point of index ν is given by:

$$\Pr[x] = P(\nu) = \frac{1}{n' \cdot n(\nu)}$$

for $\nu_1 \leq \nu \leq \nu_2$ and $P(\nu) = 0$ otherwise. The number of integers $x \in B(p,q)$ such that $\Pr[x] = 0$ is then at most:

$$(\nu_1 + \nu_3 - \nu_2 + 2) \cdot \left(\frac{j(p,q)}{q} + 1 \right) \leq N^{6\alpha} \cdot \frac{j(p,q)}{q} \tag{12}$$

For all $\nu_1 \leq \nu \leq \nu_2$, we have using (4), (5), (9), (10), (11) and $N^\alpha \geq 3$:

$$\left| P(\nu) - \frac{1}{b(p,q)} \right| \leq 10 \cdot \frac{N}{h \cdot j(p,q)} \cdot N^{-3\alpha} \tag{13}$$

Eventually, the statistical distance from the uniform distribution is:

$$\delta_G = \frac{1}{2} \sum_{x \in B(p,q)} \left| \Pr[x] - \frac{1}{b(p,q)} \right|$$

and we obtain using (11), (12) and (13):

$$\delta_G \leq 7 \cdot N^{-3\alpha}$$

B Proof of lemma 5

Let us denote $q_m = N^{1/3-4\alpha}$. For $q \geq q_m$, the probability to generate $p/q \in \mathcal{F}_k$ using $\mathcal{V}$ is $j(p,q)/|\mathbb{Z}_N^+|$. Moreover, using lemma 2, the probability that $\mathcal{V}$ generates $\perp$ is at most:

$$\Pr[\perp] = \sum_{\mathcal{F}_k | q < q_m} \frac{2 \cdot j(p,q)}{N+1} \leq 3\frac{q_m}{k} \leq 3 \cdot N^{-3\alpha} \tag{14}$$

Consequently, the statistical distance δ_V between $\mathcal{D}$ and the distribution induced by $\mathcal{V}$ is at most:

$$\delta_V = \frac{1}{2} \sum_{\mathcal{F}_k | q \geq q_m} \left| \frac{2 \cdot j(p,q)}{N+1} - \frac{b(p,q)}{b} \right| + \frac{1}{2} \Pr[\perp] + \frac{1}{2} \sum_{\mathcal{F}_k | q < q_m} \frac{b(p,q)}{b} \tag{15}$$

Let ℓ be the size of N in bits. From lemma 1, we obtain for $\ell \geq 64$:

$$|b - \frac{h}{2}| \leq 4 \cdot \ell \cdot 2^{\ell/2} \leq \frac{1}{2} \cdot N^{2/3} \leq \frac{h}{2} \cdot N^{-3\alpha} \tag{16}$$

For $q \geq q_m$, we obtain from Lemma 4 and (16):

$$\left| \frac{b(p,q)}{b} - \frac{2 \cdot j(p,q)}{N+1} \right| \leq \frac{12 \cdot j(p,q)}{N+1} \cdot N^{-3\alpha} \tag{17}$$

This gives:

$$\sum_{\mathcal{F}_k | q<q_m} \frac{b(p,q)}{b} = 1 - \sum_{\mathcal{F}_k | q \geq q_m} \frac{b(p,q)}{b} \leq 1 - (1 - 6 \cdot N^{-3\alpha}) \cdot \sum_{\mathcal{F}_k | q \geq q_m} \frac{2 \cdot j(p,q)}{N+1}$$

From (14) and using:

$$\sum_{\mathcal{F}_k} \frac{2 \cdot j(p,q)}{N+1} = 1$$

we obtain:

$$\sum_{\mathcal{F}_k | q<q_m} \frac{b(p,q)}{b} \leq 9 \cdot N^{-3\alpha} \tag{18}$$

From equation (15) and inequalities (14), (17) and (18), we obtain:

$$\delta_V \leq 9 \cdot N^{-3\alpha}$$

C Proof of theorem 3

The generator $\mathcal{G}$ combines the generators $\mathcal{V}$ and $\mathcal{G}(p,q)$. Moreover, $\mathcal{V}$ generates $p/q \in \mathcal{F}_k$ such that the statistical distance δ_G of the distribution induced by $\mathcal{G}(p,q)$ from the uniform distribution in $B(p,q)$ is at most $7 \cdot N^{-3\alpha}$. Therefore the statistical distance δ of $\mathcal{G}$ from the uniform distribution in B is at most:

$$\delta \leq \delta_V + \delta_G \leq 16 \cdot N^{-3\varepsilon/13}$$

Merkle-Damgård Revisited : how to Construct a Hash Function

Proceedings of Crypto 2005

Jean-Sébastien Coron[1], Yevgeniy Dodis*[2], Cécile Malinaud[3], and Prashant Puniya**[2]

[1] University of Luxembourg, coron@clipper.ens.fr
[2] New-York University, {dodis,puniya}@cs.nyu.edu
[3] Gemplus Card International, cecile.malinaud@normalesup.org

Abstract. The most common way of constructing a hash function (*e.g.,* SHA-1) is to iterate a compression function on the input message. The compression function is usually designed from scratch or made out of a block-cipher. In this paper, we introduce a new security notion for hash-functions, stronger than collision-resistance. Under this notion, the arbitrary length hash function H must behave as a random oracle when the fixed-length building block is viewed as a random oracle or an ideal block-cipher. The key property is that if a particular construction meets this definition, then any cryptosystem proven secure assuming H is a random oracle remains secure if one plugs in this construction (still assuming that the underlying fixed-length primitive is ideal). In this paper, we show that the current design principle behind hash functions such as SHA-1 and MD5 — the (strengthened) Merkle-Damgård transformation — does not satisfy this security notion. We provide several constructions that provably satisfy this notion; those new constructions introduce minimal changes to the plain Merkle-Damgård construction and are easily implementable in practice.

1 Introduction

Random Oracle Methodology. The random oracle model has been introduced by Bellare and Rogaway as a "paradigm for designing efficient protocols" [4]. It assumes that all parties, including the adversary, have access to a public, truly random hash function H. This model has been proven extremely useful for designing simple, efficient and highly practical solutions for many problems. From a theoretical perspective, it is clear that a security proof in the random oracle model is only a heuristic indication of the security of the system when instantiated with a particular hash function, such as SHA-1 [16] or MD5 [18]. In fact, many recent "separation" results [11, 26, 19, 2, 12, 15] illustrated various cryptographic systems secure in the random oracle model but completely insecure for *any* concrete instantiation of the random oracle (even by a *family* of hash functions). Nevertheless, these important separation results do not seem to directly attack any of the concrete, widely used cryptosystems (such as OAEP [6] and PSS [5] as used in the PKCS #1 v2.1 standard [27]) which rely on "secure hash functions". Moreover, we hope that such particular systems are in fact *secure when instantiated with a "good" hash function.* In the random oracle model, instead of making a highly non-standard (and possibly unsubstantiated) assumption that "my system is secure with this H" (e.g., H being SHA-1), one proves that the system is at least secure with an "ideal" hash function H (under standard assumptions). Such formal proof in the random oracle model is believed to indicate that

* Supported by NSF CAREER Award CCR-0133806 and TC Grant No. CCR-0311095.
** Supported by NSF Cybertrust/DARPA Grant No. CNS-0430425.

there are no structural flaws in the design of the system, and thus one can heuristically hope that no such flaws will suddenly appear with a particular, "well designed" function H. *But can we say anything about the lack of structural flaws in the design of H itself?*

BUILDING RANDOM ORACLES. On the first glance, it appears that nothing theoretically meaningful can be said about this question. Namely, we know that mathematically a concrete function H is not a random oracle, so to prove that H is "good" we need to directly argue the security of our system with this given H. And the latter task is usually unmanageable given our current tools (e.g., "realizable" properties of H such as collision-resistance, pseudorandomness or one-wayness are usually not enough to prove the security of the system). However, we argue that there is a significant gap in this reasoning. Indeed, most systems abstractly model H as a function from $\{0,1\}^*$ to $\{0,1\}^n$ (where n is proportional to the security parameter), so that H can be used on some arbitrary input domain. On the other hand, in practice such *arbitrary-length* hash functions are built by first heuristically constructing a *fixed-length* building block, such as a fixed-length compression function or a block cipher, and then iterating this building block in some manner to extend the input domain arbitrarily. For example, SHA-1, MD5, as well as all the other hash function we know of, are constructed by applying some variant of the Merkle-Damgård construction to an underlying compression function $f : \{0,1\}^{n+\kappa} \to \{0,1\}^n$ (see Figure 5):

Function $H(m_1, \ldots, m_\ell)$:
  let $y_0 = 0^n$ (more generally, some fixed IV value can be used)
  for $i = 1$ to ℓ do $y_i \leftarrow f(y_{i-1}, m_i)$
  return y_ℓ

When the number of κ-bit message blocks ℓ is not fixed, one essentially appends an extra block $m_{\ell+1}$ containing the binary representation $\langle |m| \rangle$ of the length of the message (prepended by 1 and a string of 0's in order to make everything a multiple of κ; the exact details will not matter for our discussion).

The fixed-length compression function f can either be constructed from scratch or made out of a block-cipher E via the Davies-Meyer construction (see [31] and Figure 9): $f(x, y) = E_y(x) \oplus x$. For example, the SHA-1 compression function was designed specifically for hashing, but a block-cipher can nevertheless be derived from it, as illustrated in [20].

OUR MAIN QUESTION. Given such particular and "structured" design of our hash function H,— which is actually the design used in practice,— we argue that there exists a missing link in the claim that no structural flaws exist in the design of our system. Indeed, we only know that no such flaws exist when H was modelled as a "monolithic" random oracle, and not as an iterated hash function built from some smaller building block. As since the real implementation of H as an iterated hash function has much more structure than a random monolithic hash function would have, maybe this structure could somehow invalidate the security proof in the random oracle model? To put this into a different perspective, all the ad-hoc (and hopefully "secure") design effort for widely used hash functions, such as SHA-1 and MD5, has been placed into the design of the fixed-length building block f (or E). On the other hand, even if f (or E) were assumed to be ideal, the current proofs in the random oracle model do not guarantee the security of the resulting system when such iterated hash function H is used!

Let us illustrate our point on a well known example. A common suggestion to construct a MAC algorithm is to simply include a secret key k as part of the input of the hash function, and take for example $\mathrm{MAC}(k, m) = H(k\|m)$. It is easy to see that this construction is secure when H is modelled as a random oracle [4], as no adversary can output a MAC forgery except with negligible probability. However, this MAC scheme is completely insecure for any Merkle-Damgård construction considered so far (including Merkle-Damgård strengthening used in current hash functions such as SHA-1, and any of the 64 block-cipher based variants of iterative hash-functions considered in [29, 9]), no matter which (ideal) compression function f (or a block cipher E) is used. Namely, given $\mathrm{MAC}(k, m) = H(k\|m)$, one can extend the message m with any single arbitrary block y and deduce $\mathrm{MAC}(k, m\|y) = H(k\|m\|y)$ without knowing the secret key k (even with Merkle-Damgård strengthening, one could still forge the MAC by more or less setting $y = \langle|m|\rangle$, where the actual block depends on the exact details of the strengthening). This (well known) example illustrates that the construction of a MAC from an iterated hash function requires a specific analysis, and cannot be derived from the security of this MAC with a monolithic hash function H. On the other hand, while the Merkle-Damgård transformation and its variants have been intensively studied for many "realizable" properties such as collision-resistance [13, 25, 29, 9], pseudorandomness [8], unforgeability [1, 24] and randomness extraction [14], it is clear that these analyses are insufficient to argue its applicability for the purposes of building a hash function which can be modeled as a random oracle, since the latter is a considerably stronger security notion (in fact unrealizable in the standard model). For a simple concrete example, the Merkle-Damgård strengthening is easily seen to preserve collision-resistance when instantiated with a collision-resistant compression function, while we just saw that it does not work to yield a random oracle or even just a variable-length MAC, and this holds even if the underlying compression function is modelled as a random oracle.

Our Goals. Summarizing the above discussion, our goal is two-fold. First, we would like to give a formal *definition* of what it means to implement an arbitrary-length random oracle H from a fixed-length building block f or E. The key property of this definition should be the fact that if a particular construction of H from f (or E) meets this definition, then *any application* proven secure assuming H is a random oracle would remain secure if we plug in our construction (although still assuming that the underlying fixed-length primitive f or E was ideal). In other words, we can safely use our implementation of H as if we were using a monolithic random oracle H. We remark that this means that our definition should not just preserve the pseudorandomness properties of H, but also all the other "tricks" present in the random oracle model, such as "programmability" and "extractability". For example, we could try to set $H(x) = f(h(x))$, where f is a fixed-length random oracle and h is a collision-resistant hash function (not viewed as a random oracle). While pseudorandom, this simple implementation is clearly not "extractable": for example, given output $z = f(h(x))$ for some unknown x, we can only "extract" the value $h(x)$ (by observing the random oracle queries made to f), but then have no way of extracting x itself from $h(x)$ (indeed, we will show a direct attack on this implementation in Section 3.1). This shows that the security definition we need is an interesting and non-trivial task of its own, especially if we also want it to be simple, natural and easy to use.

Second, while the definition we seek should not be too specific to some variant of the Merkle-Damgård transformation, we would like to give secure constructions which resemble what is done in practice as much as possible. Unfortunately, we already argued that the current design principle behind hash functions such as SHA-1 and MD5 — the (strengthened) Merkle-Damgård transformation — will *not* be secure for our ambitious goal. Therefore, instead of giving new and practically unmotivated constructions, our secondary goal is to come up with *minimal* and *easily implementable in practice* changes to the plain Merkle-Damgård construction, which would satisfy our security definition.

OUR RESULTS. First, we give a satisfactory definition of what it means to implement an arbitrary-length random oracle H from a fixed length primitive g (where g is either an ideal compression function f, or a an ideal block cipher E). Our definition is based on the indifferentiability framework of Maurer et al. [23]. This framework enjoys the desired closure property we seek, and is very intuitive and easy to state.

Having a good security definition, we provide several provable constructions. We start by giving three modifications to the (insecure) plain Merkle-Damgård construction which yield a secure random oracle H taking *arbitrary-length* input, from a compression function viewed as a random oracle taking *fixed-length* input. This result can be viewed as a secure *domain extender* for the random oracle, which is an interesting result of independent interest. We remark that domain extenders are well studied for such primitives as collision-resistant hash functions [13, 25], pseudorandom functions [8], MACs [1, 24] and universal one-way hash functions [7, 30]. Although the above works also showed that some variants of Merkle-Damgård yield secure domain extenders for the corresponding primitive in question, these results are not sufficient to claim a domain extender for the random oracle.

Our secure modifications to the plain Merkle-Damgård construction are the following. (1) *Prefix-Free Encoding* : we show that if the inputs to the plain MD construction are guaranteed to be *prefix-free*, then the plain MD construction is secure. (2) *Dropping Some Output Bits* : we show that by dropping a non-trivial number of output bits from the plain MD chaining, we get a secure random oracle H even if the input is not encoded in the prefix-free manner. (3) *Using NMAC construction* (see Figure 8a): we show that by applying an independent hash function g to the output of the plain MD chaining (as in the NMAC construction [8]), then once again we get a secure construction of an arbitrary-length random oracle H, in the random oracle model for f and g. (4) *Using HMAC Construction* (see Figure 8b): we show a slightly modified variant of the NMAC construction allowing us to conveniently build the function g from the compression function f itself (as in [8] when going from NMAC to HMAC)! In this latter variant, one implements a secure hash function H by making two *black-box calls to the plain Merkle-Damgård construction* (with the same fixed IV and a given compression function f): first on $(\ell+1)$-block input $0^{\kappa}m_1 \ldots m_\ell$, getting an n-bit output y, and then on one-block κ-bit input y' (obtained by either truncating or padding y depending on whether or not $\kappa > n$), getting the final output.

However, in practice most hash-function constructions are block-cipher based, either explicitly as in [29] or implicitly as for SHA-1. Therefore, we consider the question of designing an arbitrary-length random oracle H from an ideal block cipher E, specifically concentrating on using the Merkle-Damgård construction with the Davies-Meyer compres-

sion function $f(x,y) = E_y(x) \oplus x$, since this is the most practically relevant construction. We show that all of the four fixes to the plain MD chaining which worked when f was a fixed-length random oracle, are still secure (in the ideal cipher model) when we plug in $f(x,y) = E_y(x) \oplus x$ instead. Specifically, we can either use a prefix-free encoding, or drop a non-trivial number of output bits (when possible), or apply an independent random oracle g to the output of plain MD chaining, or use the optimized HMAC construction which allows us to build this function g from the ideal cipher itself.

2 Definitions

In this section, we introduce the main notations and definitions used throughout the paper. Our security notion for secure hash-function is based on the notion of indifferentiability of systems, introduced by Maurer *et al.* in [23]. This is an extension of the classical notion of indistinguishability, when one or more oracles are publicly available, such as random oracles or ideal ciphers. This notion is based on ideas from the Universal Composition framework introduced by Canetti in [10] and on the model of Pfitzmann and Waidner [28]. The indifferentiability notion in [23] is given in the framework of random systems providing interfaces to other systems, but equivalently we use this notion in the framework of Interactive Turing Machines (as in [10]).

We define an *ideal primitive* as an algorithmic entity which receives inputs from one of the parties and deliver its output immediately to the querying party. The ideal primitives that we consider in this paper are random oracles and ideal ciphers. A *random oracle* [4] is an ideal primitive which provides a random output for each new query. Identical input queries are given the same answer. An *ideal cipher* is an ideal primitive that models a random block-cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. Each key $k \in \{0,1\}^\kappa$ defines a random permutation $E_k = E(k,\cdot)$ on $\{0,1\}^n$. The ideal primitive provides oracle access to E and E^{-1}; that is, on query $(0,k,m)$, the primitive answers $c = E_k(m)$, and on query $(1,k,c)$, the primitive answers m such that $c = E_k(m)$.

We now proceed to the definition of indifferentiability [23] :

Definition 1. *A Turing machine C with oracle access to an ideal primitive $\mathcal{G}$ is said to be $(t_D, t_S, q, \varepsilon)$ indifferentiable from an ideal primitive $\mathcal{F}$ if there exists a simulator S, such that for any distinguisher D it holds that :*

$$\left|\Pr\left[D^{C,\mathcal{G}} = 1\right] - \Pr\left[D^{\mathcal{F},S} = 1\right]\right| < \varepsilon$$

The simulator has oracle access to $\mathcal{F}$ and runs in time at most t_S. The distinguisher runs in time at most t_D and makes at most q queries. Similarly, $C^{\mathcal{G}}$ is said to be (computationally) indifferentiable from $\mathcal{F}$ if ε is a negligible function of the security parameter k (for polynomially bounded t_D and t_S).

As illustrated in Figure 1, the role of the simulator is to simulate the ideal primitive $\mathcal{G}$ so that no distinguisher can tell whether it is interacting with C and $\mathcal{G}$, or with $\mathcal{F}$ and S; in other words, the output of S should look "consistent" with what the distinguisher can obtain from $\mathcal{F}$. Note that the simulator does not see the distinguisher's queries to $\mathcal{F}$; however, it can call $\mathcal{F}$ directly when needed for the simulation.

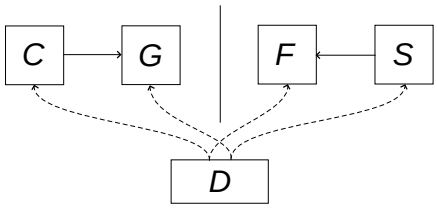

Fig. 1. The indifferentiability notion: the distinguisher D either interacts with algorithm C and ideal primitive $\mathcal{G}$, or with ideal primitive $\mathcal{F}$ and simulator S. Algorithm C has oracle access to $\mathcal{G}$, while simulator S has oracle access to $\mathcal{F}$.

In the rest of the paper, the algorithm C will represent the construction of an iterative hash-function (such as the Merkle-Damgård construction recalled in the introduction). The ideal primitive $\mathcal{G}$ will represent the underlying primitive used to build the hash-function. $\mathcal{G}$ will be either a random oracle (when the compression function is modelled as a random oracle), or an ideal block-cipher (when the compression function is based on a block-cipher). The ideal primitive $\mathcal{F}$ will represent the random oracle that the construction C should emulate. Therefore, one obtains the following setting : the distinguisher has oracle access to both the block-cipher and the hash-function, and these oracles are implemented in one of the following two ways: either the block-cipher E is chosen at random and the hash-function C is constructed from it, or the hash-function H is chosen at random and the block-cipher is implemented by a simulator S with oracle access to H. Those two cases should be indistinguishable, that is the distinguisher should not be able to tell whether the block-cipher was chosen at random and the iterated hash-function constructed from it, or the hash-function was chosen at random and the block-cipher then "tailored" to match that hash-function.

It is shown in [23] that if $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$, then $C^{\mathcal{G}}$ can replace $\mathcal{F}$ in any cryptosystem, and the resulting cryptosystem is at least as secure in the $\mathcal{G}$ model as in the $\mathcal{F}$ model. For example, if a block-cipher based iterative hash function is indifferentiable from a random oracle in the ideal cipher model, then the iterative hash-function can replace the random oracle in any cryptosystem, and the resulting cryptosystem remains secure in the ideal cipher model if the original scheme was secure in the random oracle model.

We use the definition of [23] to specify what it means for a cryptosystem to be at least as secure in the $\mathcal{G}$ model as in the $\mathcal{F}$ model. A cryptosystem is modelled as an Interactive Turing Machine with an interface to an adversary $\mathcal{A}$ and to a public oracle. The cryptosystem is run by an *environment* $\mathcal{E}$ which provides a binary output and also runs the adversary. In the $\mathcal{G}$ model, cryptosystem $\mathcal{P}$ has oracle access to C whereas attacker $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{A}$ have oracle access to $\mathcal{F}$. The definition is illustrated in Figure 2.

Definition 2. *A cryptosystem is said to be at least as secure in the $\mathcal{G}$ model with algorithm C as in the $\mathcal{F}$ model, if for any environment $\mathcal{E}$ and any attacker $\mathcal{A}$ in the $\mathcal{G}$ model, there exists an attacker $\mathcal{A}'$ in the $\mathcal{F}$ model, such that*

$$\left|\Pr\left[\mathcal{E}(\mathcal{P}^{C}, \mathcal{A}^{\mathcal{G}}) = 1\right] - \Pr\left[\mathcal{E}(\mathcal{P}^{\mathcal{F}}, \mathcal{A}'^{\mathcal{F}}) = 1\right]\right|$$

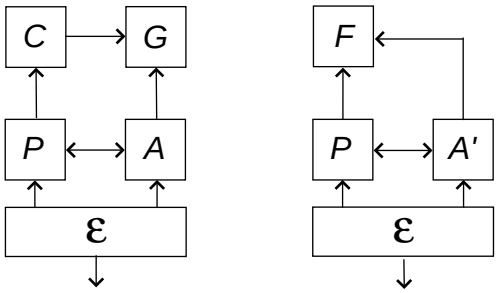

Fig. 2. The environment $\mathcal{E}$ interacts with cryptosystem $\mathcal{P}$ and attacker $\mathcal{A}$. In the $\mathcal{G}$ model (left), $\mathcal{P}$ has oracle access to C whereas $\mathcal{A}$ has oracle access to $\mathcal{G}$. In the $\mathcal{F}$ model, both $\mathcal{P}$ and $\mathcal{A}'$ have oracle access to $\mathcal{F}$

is a negligible function of the security parameter k. Similarly, a cryptosystem is said to be computationally at least as secure, etc., if $\mathcal{E}$, $\mathcal{A}$ and $\mathcal{A}'$ are polynomial-time in k.

The following theorem from [23] shows that security is preserved when replacing an ideal primitive by an indifferentiable one :

Theorem 1. *Let $\mathcal{P}$ be a cryptosystem with oracle access to an ideal primitive $\mathcal{F}$. Let C be an algorithm such that $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$. Then cryptosystem $\mathcal{P}$ is at least as secure in the $\mathcal{G}$ model with algorithm C as in the $\mathcal{F}$ model.*

Proof. We only provide a proof sketch; see [23] for a full proof. Let $\mathcal{P}$ be any cryptosystem, modelled as an Interactive Turing Machine. Let $\mathcal{E}$ be any environment, and $\mathcal{A}$ be any attacker in the $\mathcal{G}$ model. In the $\mathcal{G}$ model, $\mathcal{P}$ has oracle access to C whereas $\mathcal{A}$ has oracle access to ideal primitive $\mathcal{G}$; moreover environment $\mathcal{E}$ interacts with both $\mathcal{P}$ and $\mathcal{A}$. This is illustrated in Figure 3 (left part).

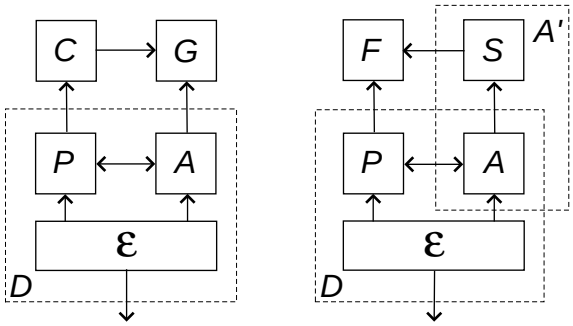

Fig. 3. Construction of attacker $\mathcal{A}'$ from attacker $\mathcal{A}$ and simulator $\mathcal{S}$.

Since $C^{\mathcal{G}}$ is indifferentiable from $\mathcal{F}$ (see Figure 1), one can replace $(C, \mathcal{G})$ by $(\mathcal{F}, S)$ with only a negligible modification of the environment's output distribution. As illustrated in

Figure 3, by merging attacker $\mathcal{A}$ and simulator $\mathcal{S}$, one obtains an attacker $\mathcal{A}'$ in the $\mathcal{F}$ model, and the difference in $\mathcal{E}$'s output distribution is negligible. □

3 Domain Extension for Random Oracles

In this section, we show how to construct an iterative hash-function indifferentiable from a random oracle, from a compression function viewed as a random oracle. We start with two simple and intuitive constructions that do not work.

3.1 $H(x) = f(h(x))$ for Random Oracle f and Collision-Resistant One-way Hash-function h

One could hope to emulate a random oracle (with arbitrary-length input) by taking :

$$C^f(x) = f(h(x))$$

where $f : \{0,1\}^n \to \{0,1\}^n$ is modelled as a random oracle and $h : \{0,1\}^* \to \{0,1\}^n$ is any collision-resistant one-way hash-function (not modelled as a random oracle). However, we show that such C^f is not indifferentiable from a random oracle; namely, we construct a distinguisher that can fool any simulator.

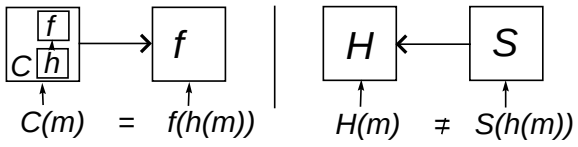

Fig. 4. The simulator cannot output $H(m)$ since it only receives $h(m)$ and cannot recover m from $h(m)$.

As illustrated in Figure 4, the distinguisher first generates an arbitrary m and computes $u = h(m)$. Then it queries $v = f(u)$ to random oracle f and queries $z = C^f(m)$ to C^f. It then checks that $z = v$ and outputs 1 in this case, and 0 otherwise. It is easy to see that the distinguisher always output 1 when interacting with C^f and f, but outputs 0 with overwhelming probability when interacting with H and any simulator S. Namely, when the distinguisher interacts with H and S, the simulator only receives $u = h(m)$; therefore, in order to output v such that $v = H(m)$, the simulator must either recover m from $h(m)$ (and then query $H(m)$) or guess the value of $H(m)$, which can be done with only negligible probability.

3.2 Plain Merkle-Damgård Construction

We show that the plain Merkle-Damgård construction (see Figure 5) fails to emulate a random oracle (taking arbitrary-length input) when the compression function f is viewed as a random oracle (taking fixed-length input). For simplicity, we only consider the usual Merkle-Damgård variant, although the discussion easily extends to the strengthened variant which appends the message length $\langle |m| \rangle$ at the last block :

Function $\mathrm{MD}^f(m_1, \ldots, m_\ell)$:
let $y_0 = 0^n$ (more generally, some fixed IV value can be used)
for $i = 1$ to ℓ do $y_i \leftarrow f(y_{i-1}, m_i)$
return $y_\ell \in \{0,1\}^n$.

where for all i, $|m_i| = \kappa$ and $f : \{0,1\}^{n+\kappa} \to \{0,1\}^n$.

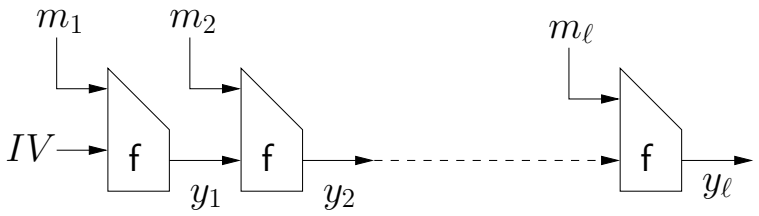

Fig. 5. The plain Merkle-Damgård Construction

We have already mentioned in introduction a counter-example based on MAC. Namely, we showed that $\mathrm{MAC}(k, m) = H(k\|m)$ provides a secure MAC in the random oracle model for H, but is completely insecure when H is replaced by the previous Merkle-Damgård construction MD^f, because of the message extension attack. In the following, we give a more direct refutation based on the definition of indifferentiability, using again the message extension attack.

We consider only one-block messages or two-block messages. For such messages, we have that $\mathrm{MD}^f(m_1) = f(0, m_1)$ and $\mathrm{MD}^f(m_1, m_2) = f(f(0, m_1), m_2)$. We build a distinguisher that can fool any simulator as follows. The distinguisher first makes a MD^f-query for m_1 and receives $u = \mathrm{MD}^f(m_1)$. Then it makes a query for $v = f(u, m_2)$ to random oracle f. The distinguisher then makes a MD^f-query for (m_1, m_2) and eventually checks that $v = \mathrm{MD}^f(m_1, m_2)$; in this case it outputs 1, and 0 otherwise. It is easy to see that the distinguisher always outputs 1 when interacting with MD^f and f. However, when the distinguisher interacts with H and S (who must simulate f), we observe that S has no information about m_1 (because S does not see the distinguisher's H-queries). Therefore, the simulator cannot answer v such that $v = H(m_1, m_2)$, except with negligible probability.

3.3 Prefix-free Merkle-Damgård

In this section, we show that if the inputs to the plain MD construction are guaranteed to be prefix-free, then the plain MD construction is secure. Namely, prefix-free encoding enables to eliminate the message expansion attack described previously. This "fix" is similar to the fix for the CBC-MAC [3], which is also insecure in its plain form. Thus, the plain MD construction can be safely used for any application of the random oracle H where the length of the inputs is fixed or where one uses domain separation (e.g., prepending $0, 1, \ldots$ to differentiate between inputs from different domains). For other applications, one must specifically ensure that prefix-freeness is satisfied.

A prefix-free code over the alphabet $\{0,1\}^\kappa$ is an efficiently computable injective function $g : \{0,1\}^* \to (\{0,1\}^\kappa)^*$ such that for all $x \neq y$, $g(x)$ is not a prefix of $g(y)$. Moreover,

it must be easy to recover x given only $g(x)$. We provide two examples of prefix-free encodings. The first one consists in prepending the message size in bits as the first block. The last block is then padded with the bit one followed by zeroes.

Function $g_1(m)$:
let N be the message length of m in bits.
write m as $(m_1, \ldots, m_\ell)$ where for all i, $|m_i| = \kappa$
and with the last block m_ℓ padded with 10^r.
let $g_1(m) = (\langle N \rangle, m_1, \ldots, m_\ell)$ where $\langle N \rangle$ is a κ-bit binary encoding of N.

An important drawback of this encoding is that the message length must be known in advance; this can be a problem for streaming applications in which a large message must be processed on the fly. Our second encoding g_2 does not suffer from this drawback, but requires to waste one bit per block of the message :

Function $g_2(m)$:
write m as $(m_1, \ldots, m_\ell)$ where for all i, $|m_i| = \kappa - 1$
and with the last block m_ℓ padded with 10^r.
let $g_2(m) = (0|m_1, \ldots, 0|m_{\ell-1}, 1|m_\ell)$.

Given any prefix-free encoding g, we consider the following construction of the iterative hash-function $\text{pf-MD}_g^f : \{0,1\}^* \to \{0,1\}^n$, using the Merkle-Damgård hash-function $\text{MD}^f : (\{0,1\}^\kappa)^* \to \{0,1\}^n$ defined previously.

Function $\text{pf-MD}_g^f(m)$:
let $g(m) = (m_1, \ldots, m_\ell)$
$y \leftarrow \text{MD}^f(m_1, \ldots, m_\ell)$
return y

Theorem 2. *The previous construction is (t_D, t_S, q, ϵ)-indifferentiable from a random oracle, in the random oracle model for the compression function, for any t_D, with $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\epsilon = 2^{-n} \cdot \ell^2 \cdot \mathcal{O}(q^2)$, where ℓ is the maximum length of a query made by the distinguisher D.*

Proof. Due to lack of space, we only provide a proof sketch for a particular prefix-free encoding which has a simpler proof; the proof for any prefix-free encoding will be provided in the full version of this paper.

The particular prefix-free encoding that we consider consists in adding the message-length as part of the input of f; moreover, the index of the current block is also included as part of the input of f, so that f can be viewed as an independent random oracle for each block m_i. Specifically, we construct an iterative hash-function $C^f : (\{0,1\}^\kappa)^* \to \{0,1\}^n$ from a compression function $f : \{0,1\}^{n+\kappa+2\cdot t} \to \{0,1\}^n$ as follows :

Function $C^f(m_1, \ldots, m_\ell)$:
let $y_0 = 0^n$
for $i = 1$ to ℓ do $y_i \leftarrow f(y_{i-1}, m_i, \langle \ell \rangle, \langle i \rangle)$
return y_ℓ

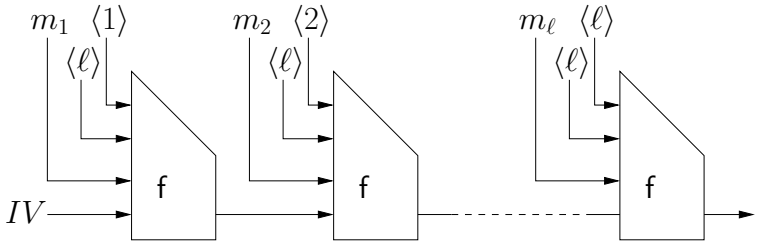

Fig. 6. Merkle-Damgård with a particular prefix-free encoding.

where for all i, $|m_i| = \kappa$. The string $\langle \ell \rangle$ is a t-bit binary encoding of the message length ℓ, and $\langle i \rangle$ is a t-bit encoding of the block index. The construction is shown in Figure 6.

In the following, we show that C^f is indifferentiable from a random oracle, in the random oracle model for f. Since the block-length ℓ is part of the input of the compression function f, we have that C^f behaves independently for messages of different length. Therefore, we can restrict ourselves to messages of fixed length ℓ, *i.e.* it suffices to show that for all ℓ, the construction C^f with message length ℓ is indifferentiable from random oracle $H_\ell : (\{0,1\}^\kappa)^\ell \to \{0,1\}^n$.

We consider for all $1 \leq j \leq \ell$ the function $C_j^f : (\{0,1\}^\kappa)^j \to \{0,1\}^n$ outputting the intermediate value y_j in C^f. From the definition of C^f, we have for all $2 \leq j \leq \ell$:

$$C_j^f(m_1, \ldots, m_j) = f(C_{j-1}^f(m_1, \ldots, m_{j-1}), m_j, \langle \ell \rangle, \langle j \rangle) \tag{1}$$

We provide a recursive proof that for all j, the construction C_j^f is indifferentiable from a random oracle. The result for C^f will follow for $j = \ell$. The property clearly holds for $j = 1$. Assuming now that it holds for $j - 1$, we show that it holds for j. We use the following lemma :

Lemma 1. *Let $h_1 : \{0,1\}^a \to \{0,1\}^n$ and $h_2 : \{0,1\}^{n+\kappa} \to \{0,1\}^n$. The construction $R^{h_1,h_2} = h_2(h_1(x), y)$ is indifferentiable from a random oracle, in the random oracle model for h_1 and h_2.*

Replacing C_{j-1}^f by h_1 and $f(\cdot, \langle \ell \rangle, \langle j \rangle)$ by h_2 in equation (1), one then obtains that C_j^f is indifferentiable from a random oracle (see Figure 7 for an illustration).

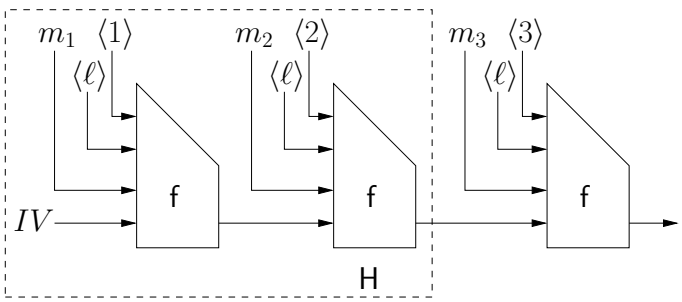

Fig. 7. The output of first two blocks is replaced by a random oracle using Lemma 1.

We now proceed to the proof of lemma 1; due to lack of space, we only provide a proof sketch. One must construct a simulator S such that interacting with $(R,(h_1,h_2))$ is indistinguishable from interacting with (H,S), where H is a random oracle. Our simulator is defined as follows :

Simulator S :

On h_1-query x, return a random $v \in \{0,1\}^n$.

On h_2-query (v',y), check if $v' = h_2(x')$ for some previously queried x'.

In this case, query (x',y) to H and output $H(x',y)$.

Otherwise return a random output.

The distinguisher either interacts with $(R,(h_1,h_2))$ or with (H,S). We denote by F the event that a collision occurs for h_1, that is $h_1(x) = h_1(x')$ for some distinct queries x, x'. We denote by F' the event that the distinguisher makes a h_2-query (v',y) such that $v' = h_1(x)$ and (x,y) was previously queried to R, but x was never queried directly to h_1 by the distinguisher. We claim that conditioned on the complement of $F \vee F'$, the simulation of S is perfect (see the full paper for a complete justification). The distinguishing probability is then at most $\Pr[F \vee F']$; for a distinguisher making at most q queries, this gives:

$$\Pr[F \vee F'] \leq \frac{2q^2}{2^n}$$

which shows a negligible distinguishing probability. □

3.4 The Chop Solution

In this section, we show that by removing a fraction of the output of the plain Merkle-Damgård construction MD^f, one obtains a construction indifferentiable from a random oracle. This "fix" is similar to the method used by Dodis et al. [14] to overcome the problem of using plain MD chaining for randomness extraction from high-entropy distributions, and to the suggestion of Lucks [22] to increase the resilience of plain MD chaining to multi-collision attacks. It is also already used in practice in the design of hash functions SHA-348 and SHA-224 [17] (both obtained by dropping some output bits from SHA-512 and SHA-256). Here we show that by dropping a non-trivial number of output bits from the plain MD chaining, one gets a secure random oracle H even if the input is not encoded in the prefix-free manner. For example, such dropping prevents the "extension" attacks we saw in the MAC application, since the attacker cannot guess the value of the dropped bits, and cannot extend the output of the MAC to a valid MAC of a longer message.

Formally, given a compression function $f : \{0,1\}^{n+\kappa} \to \{0,1\}^n$, the new construction chop-MD^f_s is defined as follows :

Function chop-$\mathrm{MD}^f_s(m)$:
  let $m = (m_1, \ldots, m_\ell)$
  $y \leftarrow \mathrm{MD}^f(m_1, \ldots, m_\ell)$
  return the first $n - s$ bits of y.

Theorem 3. *The chop-MD^f_s construction is (t_D, t_S, q, ϵ) indifferentiable from a random oracle, for any t_D, with $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\epsilon = 2^{-s} \cdot \ell^2 \cdot \mathcal{O}(q^2)$. Here ℓ is the maximum length of a query made by the distinguisher D.*

While really simple, the drawback of this method is that its exact security is proportional to $q^2 2^{-s}$, where s is the number of chopped bits and q is the number of oracle queries. Thus, to achieve adequate security level the value of s has to be relatively high, which means that short-output hash functions such as SHA-1 and MD5 cannot be fixed using this method. However, functions such as SHA-512 can naturally be fixed (say, by setting $s = 256$).

3.5 The NMAC and HMAC constructions

The NMAC construction [8], which is the basis of the popular HMAC construction, applies an *independent* hash function g to the output of the plain MD chaining. It has been shown very valuable in the design of MACs [8], and recently also randomness extractors [14]. Here we show that if g is modelled as another fixed-length random oracle *independent* from the random oracle f (used for the compression function), then once again one gets a secure construction of an arbitrary-length random oracle H, even if plain MD chaining is applied without prefix-free encoding. Intuitively, applying g gives another way to hide the output of the plain MD chaining, and thus prevent the "extension" attack described earlier.

Formally, given $f : \{0,1\}^{n+\kappa} \to \{0,1\}^n$ and $g : \{0,1\}^n \to \{0,1\}^{n'}$, the function $\mathrm{NMAC}^{f,g}$ is defined as (see Figure 8a):

Function $\mathrm{NMAC}^{f,g}(m)$:
let $m = (m_1, \ldots, m_\ell)$
$y \leftarrow MD^f(m_1, \ldots, m_\ell)$
$Y \leftarrow g(y)$
return Y

Theorem 4. *The construction $NMAC^{f,g}$ is (t_D, t_S, q, ϵ) indifferentiable from a random oracle for any t_D, $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\epsilon = 2^{-\min(n,n')}\ell^2\mathcal{O}(q^2)$, in the random oracle model for f and g, where ℓ is the maximum message length queried by the distinguisher.*

To practically instantiate this suggestion, we would like to implement f and g from a single compression function. This problem is analogous to the problem in going from NMAC to HMAC in [8], although our solution is slightly different. One simple way for achieving this is to use domain separation: e.g., by prepending 0 for calls to f and 1 — for calls to g. However, with this modeling we are effectively using the prefix-free encoding mapping $m_1 m_2 \ldots m_\ell$ to $0m_1 0m_2 \ldots 0m_\ell 10^\kappa$, which appears slightly wasteful. Additionally, this also forces us to go into the lower-level implementation details for the compression function, which we would like to avoid. Instead, our solution consists in applying two *black-box calls to the plain Merkle-Damgård construction MD^f* (with the same f and IV) : first to the input $0^\kappa m_1 \ldots m_\ell$, getting an n-bit output y, and again to κ-bit y', where y' is defined from y as follows (see Figure 8b):

Function $\mathrm{HMAC}^f(m)$:
let $m = (m_1, \ldots, m_\ell)$
let $m_0 = 0^\kappa$
$y \leftarrow \mathrm{MD}^f(m_0, m_1, \ldots, m_\ell)$
if $n < \kappa$ then $y' \leftarrow y \parallel 0^{\kappa - n}$
else $y' \leftarrow y|_\kappa$
$Y \leftarrow \mathrm{MD}^f(y')$
return Y

Intuitively, we are almost using the NMAC construction with $g(y) = f(IV, y')$ (where y' is obtained from y as above), except we prepend a fixed block $m_0 = 0^\kappa$ to our message. This latter tweak is done to ensure that there are no inter-dependencies between using the same IV on y' and the first message block (which would have been under adversarial control had we not prepended m_0). Indeed, it is very unlikely that "high-entropy" y' will ever be equal to $m_0 = 0^\kappa$, so the analysis for NMAC can be easily extended for this optimization.

Theorem 5. *The construction $HMAC^f$ is (t_D, t_S, q, ϵ) indifferentiable from a random oracle for any t_D, $t_S = \ell \cdot \mathcal{O}(q^2)$ and $\epsilon = 2^{-\min(n,\kappa)} \cdot \ell^2 \cdot \mathcal{O}(q^2)$, in the random oracle model for f, where ℓ is the maximum message length queried by the distinguisher.*

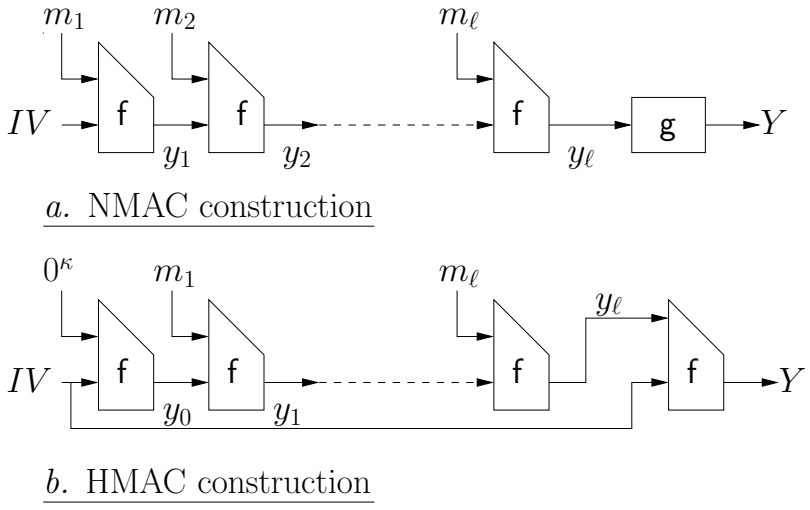

Fig. 8. The NMAC and HMAC constructions

4 Constructions using Ideal Cipher

In practice, most hash-function constructions are block-cipher based, either explicitly as in [29] or implicitly as for SHA-1. Therefore, we consider the question of designing an arbitrary-length random oracle H from an ideal block cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \rightarrow \{0,1\}^n$, specifically concentrating on using the Merkle-Damgård construction with the

Davies-Meyer compression function $f(x, y) = E_y(x) \oplus x$ (see Figure 9), since this is the most practically relevant construction. We notice that the question of designing a *collision-resistant* hash function H from an ideal block cipher was explicitly considered by Preneel, Govaerts and Vandewalle in [29], and latter formalized and extended by Black, Rogaway and Shrimpton [9]. Specifically, the authors of [9] actually considered 64 block-cipher variants of the Merkle-Damgård transform (which included the Davies-Meyer variant among them), and formally showed that exactly 20 of these variations (including the Davies-Meyer variant) are collision-resistant when the block cipher E is modeled as an ideal cipher. However, while our work will also model E as an ideal cipher, our security goal is considerably stronger than mere collision-resistance. Indeed, we already pointed out that none of the 64 variants above can withstand the "extension" attack on the MAC application, even with the Merkle-Damgård strengthening. And even when restricting to a fixed number of blocks ℓ (which invalidates the "extension" attack), collision-resistance is completely insufficient for our purposes. For example, the authors of [9] show the collision-resistance when using the plain MD chaining with fixed IV and compression function $f(x, y) = E_y(x)$. On the other hand, it is easy to see that this method does not provide a secure random oracle H according to our definition.

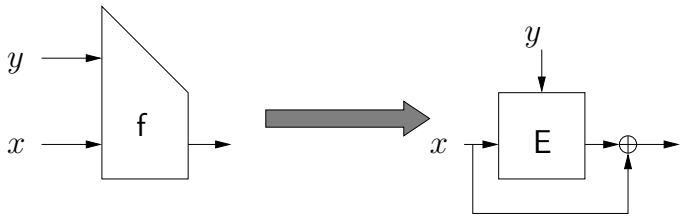

Fig. 9. The Davies-Meyer Compression function

From a different direction, if we could show that the Davies-Meyer compression function $f(x, y) = E_y(x) \oplus x$ is a secure random oracle when E is an ideal block-cipher, then we could directly apply any of the three fixes discussed above. Unfortunately, this is again not the case: intuitively, the above construction allows anybody to compute x from $f(x, y) \oplus x$ and y (since $x = E_y^{-1}(f(x, y) \oplus x)$), which should not be the case if f was a true random oracle. Thus, we need a direct proof to argue the security of the Davies-Meyer construction. Luckily, using such direct proofs we indeed argue that all of the fixes to the plain MD chaining which worked when f was a fixed-length random oracle, are still secure when $f(x, y) = E_y(x) \oplus x$ is used instead. Namely, we can either use a prefix-free encoding, or drop a non-trivial number of output bits, or apply an independent random oracle g to the output of plain MD chaining. With respect to this latter fix, we also show that we can implement this independent g using the ideal cipher itself, similarly to the case with an ideal compression function f.

Formally, given a block-cipher $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, the plain Merkle-Damgård hash-function with Davies-Meyer's compression function is defined as :

Function $\mathrm{MD}^E(m_1, \ldots, m_\ell)$:
let $y_0 = 0^n$ (more generally, some fixed IV value can be used)
for $i = 1$ to ℓ do $y_i \leftarrow E_{m_i}(y_{i-1}) \oplus y_{i-1}$
return $y_\ell \in \{0,1\}^n$.

where for all i, $|m_i| = \kappa$. The block-cipher based iterative hash-functions pf-MD_g^E, chop-MD_s^E, NMAC_g^E and HMAC^E are then defined as in section 3, using MD^E instead of MD^f. The proof of the following theorem is given in the full version of this paper.

Theorem 6. *The block-cipher based constructions pf-MD_g^E, chop-MD_s^E, $NMAC_g^E$ and $HMAC^E$ are (t_D, t_S, q, ϵ)-indifferentiable from a random oracle, in the ideal cipher model for E, for any t_D and $t_S = \ell \cdot \mathcal{O}(q^2)$, with $\epsilon = 2^{-n} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ for pf-MD_g^E, $\epsilon = 2^{-s} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ for chop-MD_s^E, $\epsilon = 2^{-\min(n,n')} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ for $NMAC_g^E$ and $\epsilon = 2^{-\min(\kappa,n)} \cdot \ell^2 \cdot \mathcal{O}(q^2)$ for $HMAC^E$. Here ℓ is the maximum message length queried by the distinguisher.*

5 Conclusion

In this paper, we pointed the attention of the cryptographic community to the gap between assuming an arbitrary-length random oracle H and assuming a fixed-length ideal building block for H such as a fixed-length compression function or a block cipher. We then provided a formal definition which suffices to eliminate this gap, noticed that the current iterative hash functions like SHA-1 and MD5 do not satisfy our security notion, and showed several practically motivated, easily implementable and provably secure fixes to the plain Merkle-Damgård transformation. Specifically, one can either ensure that all the inputs appear in the prefix-free form, or drop a nontrivial number of the output bits (if the output of the hash function is long enough to allow it), or, — when the above methods are not applicable — apply an independent fixed-length hash function to the output, which, as we illustrated, can be conveniently implemented using the corresponding building block itself.

An interesting open problem is to provide a construction in the opposite direction, that is, a construction that securely realizes an ideal block-cipher (or a random permutation) from a random oracle. One could use the Luby-Rackoff construction of a pseudo-random permutation from a pseudo-random function [21], but the major difference is that here the adversary has oracle access to the inner functions. One can show that at least six rounds are required to securely realize a random permutation from a random oracle (which should be contrasted with the secret-key case where four rounds are necessary and sufficient [21]), but we were not able to find a proof that six or more rounds would be sufficient.

Acknowledgments: We would like to deeply thank Victor Shoup for his invaluable contribution to all aspects of this work. We also thank the anonymous referees for many useful comments.

References

1. J. H. An, M. Bellare, *Constructing VIL-MACs from FIL-MACs: Message Authentication under Weakened Assumptions*, CRYPTO 1999, pages 252-269.

2. Mihir Bellare, Alexandra Boldyreva and Adriana Palacio. An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem. Proccedings of Eurocrypt 2004.
3. M. Bellare, J. Kilian, and P. Rogaway. The Security of Cipher Block Chaining. In *Crypto '94*, pages 341–358, 1994. LNCS No. 839.
4. M. Bellare and P. Rogaway, *Random oracles are practical : a paradigm for designing efficient protocols*. Proceedings of the First Annual Conference on Computer and Commmunications Security, ACM, 1993.
5. M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*. Proceedings of Eurocrypt'96, LNCS vol. 1070, Springer-Verlag, 1996, pp. 399-416.
6. M. Bellare and P. Rogaway, *Optimal Asymmetric Encryption*, Proceedings of Eurocrypt'94, LNCS vol. 950, Springer-Verlag, 1994, pp. 92–111.
7. M. Bellare and P. Rogaway, *Collision-Resistant Hashing: Towards Making UOWHFs Practical*, In *Crypto '97*, LNCS Vol. 1294.
8. M. Bellare, R. Canetti, and H. Krawczyk, *Pseudorandom Functions Re-visited: The Cascade Construction and Its Concrete Security*, In Proc. 37th FOCS, pages 514-523. IEEE, 1996.
9. J. Black, P. Rogaway, T. Shrimpton, *Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV*, in Advances in Cryptology - CRYPTO 2002, California, USA.
10. R. Canetti, *Universally Composable Security: A New Paradigm for Cryptographic Protocols*, proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS), 2001. Cryptology ePrint Archive, Report 2000/067, `http://eprint.iacr.org/`.
11. R. Canetti, O. Goldreich and S. Halevi, *The random oracle methodology, revisited*, STOC' 98, ACM, 1998.
12. Ran Canetti, Oded Goldreich and Shai Halevi. On the random oracle methodology as applied to Length-Restricted Signature Schemes. In *Proceedings of Theory of Cryptology Conference*, pp. 40–57, 2004.
13. I. Damgård, *A Design Principle for Hash Functions*, In Crypto '89, pages 416-427, 1989. LNCS No. 435.
14. Y. Dodis, R. Gennaro, J. Håstad, H. Krawczyk, and T. Rabin, *Randomness Extraction and Key Derivation Using the CBC, Cascade and HMAC Modes*, Advances in Cryptology - CRYPTO, August 2004.
15. Y. Dodis, R. Oliveira, K. Pietrzak, *On the Generic Insecurity of the Full Domain Hash*, Advances in Cryptology - CRYPTO, August 2005.
16. FIPS 180-1, *Secure hash standard*, Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, April 17 1995 (supersedes FIPS PUB 180).
17. National Institute of Standards and Technology (NIST). Secure hash standard. FIPS 180-2. August 2002.
18. RFC 1321, *The MD5 message-digest algorithm*, Internet Request for Comments 1321, R.L. Rivest, April 1992.

19. Shafi Goldwasser and Yael Tauman. On the (In)security of the Fiat-Shamir Paradigm. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science* (2003), 102-114.
20. H. Handschuh and D. Naccache, *SHACAL*, In B. Preneel, Ed., First Open NESSIE Workshop, Leuven, Belgium, November 13-14, 2000
21. M. Luby and C. Rackoff, *How to construct pseudo-random permutations from pseudo-random functions*, SIAM J. Comput., Vol. 17, No. 2, April 1988.
22. Stefan Lucks. *Design Principles for Iterated Hash Functions*, available at E-Print Archive, `http://eprint.iacr.org/2004/253`.
23. U. Maurer, R. Renner, and C. Holenstein, *Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology*, Theory of Cryptography - TCC 2004, Lecture Notes in Computer Science, Springer-Verlag, vol. 2951, pp. 21-39, Feb 2004.
24. Ueli Maurer and Johan Sjodin. *Single-key AIL-MACs from any FIL-MAC*, In *ICALP 2005*, July 2005.
25. R. Merkle, *One way hash functions and DES*, Advances in Cryptology, Proc. Crypto'89, LNCS 435, G. Brassard, Ed., Springer-Verlag, 1990, pp. 428-446.
26. Jesper Buus Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-Committing Encryption Case. In *Advances in Cryptology - Crypto 2002 Proceedings* (2002), 111 -126
27. PKCS #1 v2.1, *RSA Cryptography Standard (draft)*, document available at `www.rsasecurity.com/rsalabs/pkcs`.
28. B. Pfitzmann and M. Waidner, *A model for asynchronous reactive systems and its application to secure message transmission*. In IEEE Symposium on Security and Privacy, pages 184-200. IEEE Computer Society Press, 2001.
29. B. Preneel, R. Govaerts and J. Vandewalle, *Hash Functions Based on Block Ciphers: A Synthetic Approach*, in Advances in Cryptology - CRYPTO '93,, Santa Barbara, California, USA.
30. V. Shoup, *A composition theorem for universal one-way hash functions*, In *Eurocrypt '00*, pp. 445–452, LNCS Vol. 1807.
31. R. Winternitz, *A secure one-way hash function built from DES*, in Proceedings of the IEEE Symposium on Information Security and Privacy, pages 88-90. IEEE Press, 1984.

Résumé

Le but de la cryptographie est de construire des schémas qui accomplissent un certain objectif malgré la présence d'un attaquant. Pour formaliser la sécurité d'un schéma, il faut donc spécifier ce que l'attaquant a le droit de faire, et sous quelles conditions son attaque réussit. Le schéma sera alors considéré comme "sûr" si on peut démontrer qu'une telle attaque est impossible (sauf éventuellement avec probabilité négligeable), le plus souvent sous une certain hypothèse de complexité (par exemple, factoriser est difficile). Le domaine de la *sécurité prouvée* est la combinaison de ces trois étapes : définition, schéma, preuve. C'est cette approche qui est maintenant dominante dans la recherche en cryptographie.

Dans ce mémoire, nous commençons par rappeler comment se formalisent la sécurité d'un schéma de chiffrement à clef publique et la sécurité d'un schéma de signature. Nous proposons une preuve de sécurité améliorée (et optimale) pour le schéma de signature PSS. Nous proposons la première preuve de sécurité pour les standards de signature ISO 9796-2 et PKCS #1 v1.5. Nous introduisons aussi une nouvelle définition de sécurité pour les fonctions de hachage, plus forte que la résistance aux collisions, et nous proposons plusieurs constructions pratiques qui permettent de satisfaire cette définition.

Nous rappelons aussi les principales attaques contre certaines variantes de RSA, en particulier les attaques basées sur le théorème de Coppersmith. Ces attaques ne mettent pas en cause l'algorithme RSA lui-même, mais plutôt son utilisation incorrecte. Nous proposons aussi un algorithme plus simple que celui de Coppersmith pour trouver les petites racines d'un polynôme à deux variables sur les entiers.

Abstract

Cryptography is about making schemes that accomplish some goal despite the presence of an adversary. To formalize the security of a cryptosystem, one must therefore specify what the adversary is allowed to do, and when the attack is successful. A cryptosystem will be said "secure" if one can show that such attack is impossible (except maybe with negligible probability), under some complexity assumption (for example, factoring is hard). This approach – definition, scheme, proof – is called *provable security*, and is now mainstream in today's cryptography.

In this report, we first recall the security definitions for public-key encryption and signature. We propose an improved (and optimal) security proof for the PSS signature scheme. We propose the first security proof for the signature standards ISO 9796-2 and PKCS #1 v1.5. We also introduce a new security definition for hash functions, stronger than collision resistance, and propose various practical constructions secure under this definition.

We also recall the main attacks against some variants of the RSA cryptosystem, in particular the attacks based on Coppersmith's theorem. Those attacks do not endanger the RSA cryptosystem itself, but illustrate the danger of improper use of RSA. We also propose a simplification of Coppersmith's algorithm for finding small roots of bivariate polynomial equations over the integers.