An introduction to provable security

Dr Douglas Stebila
Queensland University of Technology

AMSI Winter School on Cryptography
July 10–11, 2014

Abstract

These notes aim to give a brief introduction to the provable security paradigm in cryptography. We will focus on game-based security definitions using reductionist proofs: a security property for a cryptographic scheme is defined as a game between a challenger and an adversary, and security is usually shown by reducing the problem of winning the game (and thereby "breaking" security) to the problem of breaking some underlying hard problem. We structure our proofs primarily using sequences of games. We will develop the techniques using three examples: the collision resistance of the Merkle–Damgård construction for hash functions, the semantic security of the basic ElGamal public key encryption scheme, and the unforgeability of the RSA Full-Domain Hash signature scheme in the random oracle model.

Contents

1 Starting off with hash functions

Definition 1 (Hash function)**.** *A* hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$ *maps arbitrary-length binary strings to binary strings of length* λ*.*

There are several properties that applications often want from hash functions:

- *Collision-resistance*: It should be hard to find two distinct inputs x and x' such that $H(x) = H(x')$.
- *Preimage resistance*: Given $y \in \{0,1\}^\lambda$, it should be hard to find a value x such that $H(x) = y$.
- *Second preimage resistance*: Given x, it should be hard to find a value $x' \neq x$ such that $H(x) = H(x')$.

To prove that a hash function has collision resistance, we need a more precise definition.

For now, we will put aside what it means to be "hard", and instead begin by trying to define the task for an attacker and measure that attacker's ability to complete the task.

1.1 Security experiment

A *security experiment* is a game played between two algorithms: a *challenger* and an *attacker*. The challenger typically does three things:

1. *Setup.* Sets up the experiment, for example by generating any parameters or long-term keys.
2. *Execution.* Executes the adversary by giving it some input, letting it run, optionally interacting with it, and receiving the output of the adversary.
3. *Winning condition.* Deciding if the adversary has completed the task ("won the game").

The challenger in the security experiment for collision resistance (coll) of a hash function H against an attacker $\mathcal{A}$ is shown in Figure 1.

$\underline{\mathbf{Exp}^{\mathsf{coll}}_H(\mathcal{A})}$

1: // *Execution:*
2: $(x, x') \xleftarrow{\$} \mathcal{A}()$
3: // *Winning condition:*
4: **if** $(x \neq x')$ and $(H(x) = H(x'))$ **then**
5: **return** 1
6: **else**
7: **return** 0
8: **end if**

Figure 1: Security experiment for collision resistance

The adversary $\mathcal{A}$ is usually considered to be a probabilistic algorithm. Formally defining algorithms is more work than we will do here, but can be done rigourously using Turing machines. A *probabilistic algorithm* A most generally is a deterministic algorithm that takes as input a value x and randomness $r \in \{0,1\}^*$ and computes an output y, denoted $y \leftarrow A(x; r)$. Often we assume that the random coins are not given explicitly, and instead are chosen uniformly at random from a set of an appropriate size ρ, in which case $y \xleftarrow{\$} A(x)$ is shorthand for $r \xleftarrow{\$} \{0,1\}^\rho \ ; \ y \leftarrow A(x; r)$.

1.2 Success probability

Once we have a security experiment, we then can measure the attacker's ability to complete the task using probability.

We define the *success probability* of algorithm $\mathcal{A}$ in finding a collision in H as

$$\mathbf{Succ}_H^{\mathsf{coll}}(\mathcal{A}) = \Pr\left(\mathbf{Exp}_H^{\mathsf{coll}}(\mathcal{A}) = 1\right) \ ,$$

where the probability is taken over the random coins used by the challenger and by $\mathcal{A}$.

1.3 Security

For a hash function to be collision-resistant, we want the success probability to be zero or small for all algorithms $\mathcal{A}$. This turns out to be a very high bar (called "information theoretically secure" or "unconditionally secure", a special case of which is "perfectly secure"). Few practical schemes are information theoretically secure. For most practical schemes, we have to consider a restricted class of attacker algorithms, such as polynomial-time algorithms or algorithms that take at most 2^{80} steps.

We will hold off quantifying over classes of adversaries for a while since it is a non-trivial task.

For hash functions, the birthday paradox tells us that there is an algorithm that finds collisions with reasonable probability in time $2^{\lambda/2}$.

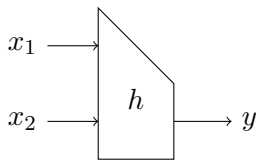
1.4 Building an arbitrary-length hash function

Cryptographers try to build bigger cryptographic schemes from other smaller cryptographic schemes.

- This allows implementers to re-use software code.
- This allows cryptographers to build secure systems out of secure building blocks.

Recall that our definition of a hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$ takes arbitrary-length inputs. It's hard to build a secure function that works for arbitrary-length inputs. It's much easier to build a secure function that takes fixed-length inputs, and then hope we can use that to build one that takes arbitrary-length inputs.

Definition 2 (Compression function)**.** *A* compression function $h : \{0,1\}^\mu \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ *maps binary strings of length* $\mu + \lambda$ *to binary strings of length* λ*.*



The *Merkle–Damgård construction*, as shown in Figure 2 builds an arbitrary-length hash function $H : \{0,1\}^{\leq 2^{64}} \to \{0,1\}^\lambda$ from a compression function $h : \{0,1\}^\mu \times \{0,1\}^\lambda \to \{0,1\}^\lambda$.

1. Split input m up into μ-bit blocks $m = m_1 \| m_2 \| \ldots \| m_\ell$, where m_ℓ may be less than μ bits long.
2. Add padding $pad = 1000\ldots0 \| L$ where L is the bit length of m represented as a 64-bit integer and we use enough zeros to pad out m_ℓ to μ bits (or overflow into the next block if not enough space in m_ℓ.
3. Input each block into compression function h along with chained output; use *initialization vector (IV)* to get started.

Here's the theorem we'd like to prove:

Theorem 1 (Collision-resistance of Merkle–Damgård construction)**.** *If h is a collision-resistant compression function, then H (the Merkle–Damgård hash function constructed from h) is collision-resistant.*

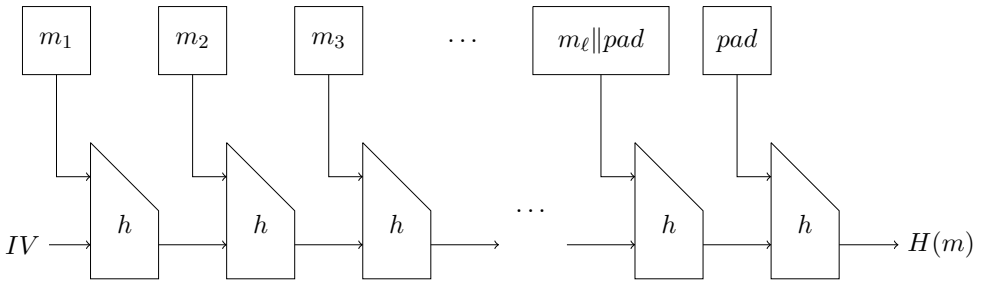


Figure 2: The Merkle–Damgård construction of hash function H from compression function h.

Since we haven't yet said what it means to be hard, we'll prove something slightly different:

Theorem 2 (Collision-resistance of Merkle–Damgård construction)**.** *Let h be a compression function and H be the Merkle–Damgård hash function constructed from h. Given a collision $M \neq M'$ for H, "we can easily find" a collision $m \neq m'$ for h.*

Proof. Let H_i and H'_i, respectively denote the intermediate hash values in the computation of $H(M)$ and $H(M')$ respectively: $H_0 = IV, H_i = h(m_i, H_{i-1})$. This constructs a chain of hash values:

$$\begin{aligned} IV = H_0 \quad , \quad H_1 \quad , \quad \ldots \quad , \quad H_t \quad , \quad H_{t+1} = H(M) \\ IV = H_0 \quad , \quad H'_1 \quad , \quad \ldots \quad , \quad H'_r \quad , \quad H'_{r+1} = H(M') \end{aligned}$$

with

$$h(M_t \| pad, H_t) = H_{t+1} = H'_{r+1} = h(M'_r \| pad', H'_r) \ .$$

If $H_t \neq H'_r$ or $M_t \neq M'_r$ or $pad \neq pad'$, then we have a collision on h, and we can stop.

Otherwise, we have that $H_t = H'_r$ and $M_t = M'_r$ and $pad = pad'$. Since $pad = pad'$ and they encode t and r respectively, we must have that $t = r$. So

$$h(M_{t-1}, H_{t-1}) = H_t = H'_t = h(M'_{t-1}, H'_{t-1}) \ .$$

If $H_{t-1} \neq H'_{t-1}$ or $M_{t-1} \neq M'_{t-1}$, then we have a collision on h, and we can stop.

Otherwise, we have that $H_{t-1} = H'_{t-1}$ and $M_t = M'_t$ and $M_{t-1} = M'_{t-1}$. Iterate all the way to the beginning, and either

- find a collision on h at some stage, or
- for all i, $M_i = M'_i$. But this implies $M = M'$, which contradicts the assumption $M \neq M'$ in the theorem.

Thus we can always "easily find" a collision for h given a collision for H. □

The *contrapositive* allows us to turn the above theorem into a security statement:

Corollary 1. *If no one can easily find a collision in the compression function h, then no one can easily find a collision in the hash function H.*

Formalizing "easily find" is sometimes challenging but we will work towards it.

Corollary 2. *Let h be a compression function, H be the Merkle–Damgård hash function constructed from h, and let $\mathcal{A}$ be an algorithm that attempts to find a collision in H. Then, for the algorithm $\mathcal{B}$ given implicitly in the proof of Theorem 2,*

$$\mathbf{Succ}^{\mathsf{coll}}_{H}(\mathcal{A}) \leq \mathbf{Succ}^{\mathsf{coll}}_{h}(\mathcal{B}^{\mathcal{A}}) \ ,$$

where $\mathcal{B}^{\mathcal{A}}$ denotes that $\mathcal{B}$ can call $\mathcal{A}$ as a subroutine.

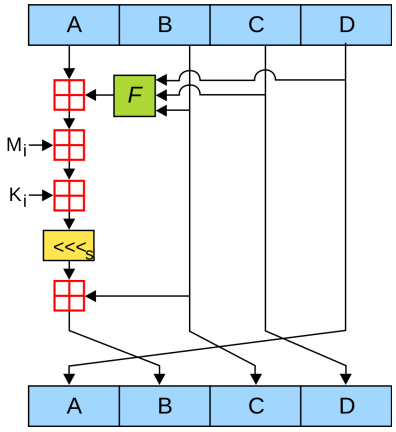


Figure 3: One MD5 operation.
MD5 consists of 64 of these operations, grouped in four rounds of 16 operations. F is a nonlinear function; one function is used in each of the 4 rounds. M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant, different for each operation. $\lll_s$ denotes a left bit rotation by s places; s varies for each operation. $\boxplus$ denotes addition modulo 2^{32}. (Image and caption from Wikipedia.)

The algorithm $\mathcal{B}$ acts as a *(Turing) reduction* from the problem of collision-finding in h to the problem of collision-finding in H using subroutine $\mathcal{A}$.

1.5 Merkle–Damgård in practice

Many real-world hash functions are built using Merkle–Damgård, such as MD5, SHA-1, and the SHA-2 family.

Here is one (out of 64) iterations comprising the MD5 compression function h_{MD5}:

Question: Is h_{MD5} collision-resistant?

Answer:

- Collisions definitely exist. The birthday paradox finds collisions in MD5 in about $2^{128/2} = 2^{64}$ operations.
- At the rump session of CRYPTO 2004, Xiaoyun Wang and coauthors gave two colliding inputs to MD5.
- The best attack at present finds collisions in 2^{18} operations. It is even possible to construct "meaningful" collisions, such as two different public key certificates or two different executable files.

That collisions can be easily found in MD5 does not invalidate the security proof of the Merkle–Damgård construction: both Theorem 2 and Corollary 2 are still valid. What's not true is the precondition of Corollary 1, that "no one can easily find a collision in the compression function" h_{MD5}.

2 Overview of provable security

(This section based in part on lecture slides by Alexander Dent.[1]*)*

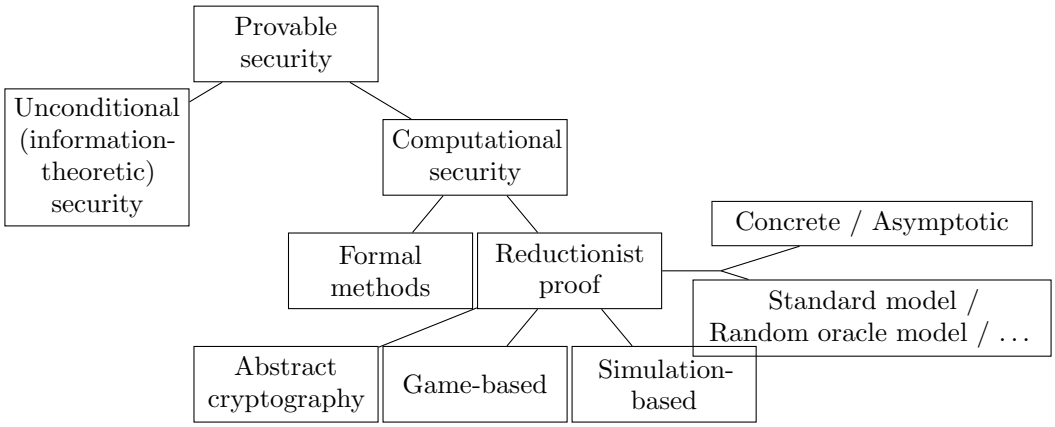
Provable security aims to show mathematically that a specific property of a scheme cannot

[1] http://www.cs.bris.ac.uk/Research/CryptographySecurity/SummerSchool2009/slides/Alex.pdf

be broken by a class of attackers. We need:

- a definition of the scheme
- a definition of security, including an execution environment / model and a winning condition
- a class of attackers
- a proof that no attacker in that class can achieve the winning condition for that scheme

Provable security has several approaches:



- *Unconditional or information-theoretic security.* Security against "all" attackers – no bound on computation. Examples: Shannon [Sha49], one-time pad, etc. Typically uses combinatorial or information theory techniques.
- *Computational security.* Security against a class of attacker algorithms. Typically uses complexity theoretic techniques.
  - *Formal methods.* Computer-verified security of scheme / protocol. Typically assumes underlying cryptography is perfect.
  - *Reductionist proof.* Manual proof of security of scheme / protocol. Typically reduces security of scheme to security of underlying hard problem. Introduced by Goldwasser and Micali [GM84].
    - *Abstract cryptography or constructive cryptography* [Mau05, Mau12]. A recent approach aiming to describe cryptographic primitives and reductions as algebraic objects, which can be combined using algebraic rules.
    - *Game-based.* Security defined as a game played between a challenger and an attacker. The challenger's behaviour defines the security model.
    - *Simulation-based.* Security defined as an attacker being unable to distinguish between interact with the real scheme and an "idealized" scheme with a simulator trying to simulate the real scheme. Specific example: *universal composability (UC)* framework [Can01].

    Reductionist security can be

    - *Asymptotic.* Class of attacker algorithms: *probabilistic polynomial time* (polynomial in a *security parameter*). Attacker's success probability / advantage: *negligible* in the security parameter, meaning smaller than the inverse of any polynomial.
    - *Concrete.* Class of attacker algorithms: probabilistic time-t algorithms. Attacker's success probability / advantage: $\leq \epsilon$.

    Reductionist proofs sometimes make use of powerful simplifying assumptions:

    - Ideal hash functions, the *random oracle model* [BR93].
    - Ideal block ciphers, the *ideal cipher model.*
    - Ideal groups, the *generic group model* [Sho97].

Reductionist proofs without such assumptions are said to be in the *standard model*. Reductionist proofs and formal methods are starting to merge with computer-verifiable reductionist proofs (see for example EasyCrypt[2]).

In Section 1, we developed a game-based definition of security for collision resistance, and used a standard model reductionist proof with concrete values to show computational security of collision resistance for the Merkle–Damgård construction.

3 Public key encryption

(This section based in part on Victor Shoup's paper "Sequences of Games: A Tool for Taming Complexity in Security Proofs" [Sho06].)

3.1 Basic definitions for public key encryption

Definition 3 (Public key encryption scheme)**.** *For a message space $\mathcal{M}$, a* public key encryption scheme Π *is a triple of algorithms* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

- $\mathsf{KeyGen}() \xrightarrow{\$} (sk, pk)$*: A probabilistic* key generation *algorithm that takes no input, and generates a secret key / public key pair.*
- $\mathsf{Enc}(pk, m) \xrightarrow{\$} c$*: A probabilistic* encryption *algorithm that takes as input a public key pk and a message $m \in \mathcal{M}$, and outputs a ciphertext c.*
- $\mathsf{Dec}(sk, c) \to m$ *or* $\perp$*: A deterministic* decryption *algorithm that takes as input a secret key sk and a ciphertext c, and outputs either a message m or a distinguished error symbol* $\perp$.

Definition 4 (Correctness)**.** *A public key encryption scheme* Π *is* correct *if for all $m \in \mathcal{M}$, all $(sk, pk) \leftarrow \mathsf{KeyGen}()$, and all $c \leftarrow \mathsf{Enc}(pk, m)$, we have that $\mathsf{Dec}(sk, c) = m$.*

Sometimes it is sufficient that valid ciphertexts decrypt correctly with high probability.

There are several possible security goals for the attacker:

- *Key recovery*: Compute sk, allowing the attacker to decrypt any message.
- *Message recovery*: Given a ciphertext, decrypt it.
- *Indistinguishability*: Given a ciphertext encrypting either m_0 or m_1, decide which.

Indistinguishability turns out to be equivalent to *semantic security*: the attacker learns nothing about the message given the ciphertext, except possibly its length. [GM84]

We can give the attacker different powers:

- *Key-only attack*: The attacker receives pk.
- *Chosen-plaintext attack (CPA)*: The attacker may encryptions of messages of his choosing.
- *Chosen-ciphertext attack (CCA1)*: The attacker may obtain decryptions of ciphertexts of his choosing (up until he receives the challenge).
- *Adaptive chosen-ciphertext attack (CCA2)*: The attacker may adaptively obtain decryptions of his choosing, even after he receives the challenge.

The "best" security notion is the one where we set the weakest goal for the attacker and give him the strongest powers, which in this case is *indistinguishability under adaptive chosen-*

[2] https://www.easycrypt.info

$\underline{\mathbf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi}(\mathcal{A})}$

1: // *Setup:*
2: $(sk, pk) \xleftarrow{\$} \mathsf{KeyGen}()$
3: // *Execution:*
4: $(m_0, m_1, st) \xleftarrow{\$} \mathcal{A}(pk)$
5: $b \xleftarrow{\$} \{0,1\}$
6: $c \xleftarrow{\$} \mathsf{Enc}(pk, m_b)$
7: $b' \xleftarrow{\$} \mathcal{A}(st, c)$
8: // *Winning condition:*
9: **if** $b = b'$ **then**
10: **return** 1
11: **else**
12: **return** 0
13: **end if**

(a) Semantic security / indistinguishability under chosen plaintext attack (IND-CPA)

$\underline{\mathbf{Exp}^{\mathsf{ind\text{-}cca}}_{\Pi}(\mathcal{A})}$

1: // *Setup:*
2: $(sk, pk) \xleftarrow{\$} \mathsf{KeyGen}()$
3: // *Execution:*
4: $(m_0, m_1, st) \xleftarrow{\$} \mathcal{A}^{\mathsf{Dec}(sk,\cdot)}(pk)$
5: $b \xleftarrow{\$} \{0,1\}$
6: $c \xleftarrow{\$} \mathsf{Enc}(pk, m_b)$
7: $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{Dec}(sk,\cdot \neq c)}(st, c)$
8: // *Winning condition:*
9: **if** $b = b'$ **then**
10: **return** 1
11: **else**
12: **return** 0
13: **end if**

(b) Indistinguishability under adaptive chosen ciphertext attack (IND-CCA).

Figure 4: Security experiments for public key encryption.

ciphertext attack (IND-CCA2). The relationships between these and other security notions for public key encryption have been explored [BDPR98].

We will focus on IND-CPA, the full definition of which is shown in Figure 4a.

Notice that $\mathcal{A}$ can guess b with probability $1/2$ trivially. We are concerned with how much better $\mathcal{A}$ can do than that, so we are interested in $\mathcal{A}$'s *advantage* in winning the ind-cpa experiment for Π:

$$\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi}(\mathcal{A}) = \left| \Pr\left(\mathbf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Pi}(\mathcal{A}) = 1\right) - \frac{1}{2} \right| \ .$$

3.2 Basic ElGamal encryption scheme

Let G be a cyclic group of prime order q and let g be a generator of G. The basic ElGamal public key encryption scheme [ElG84], denoted $\mathsf{ElGamal}(G, g, q)$, is shown in Figure 5 below.

$\underline{\mathsf{KeyGen}()}$

1: $x \xleftarrow{\$} \mathbb{Z}_q$
2: $X \leftarrow g^x$
3: **return** $(sk, pk) \leftarrow (x, X)$

$\underline{\mathsf{Enc}(pk = X, m)}$

1: $y \xleftarrow{\$} \mathbb{Z}_q$
2: $c_1 \leftarrow g^y$
3: $Z \leftarrow X^y$
4: $c_2 \leftarrow m \cdot Z$
5: **return** $c \leftarrow (c_1, c_2)$

$\underline{\mathsf{Dec}(sk = x, c)}$

1: **return** $m \leftarrow c_2 / c_1^x$

Figure 5: Basic ElGamal encryption scheme algorithms.

Exercise 1. Verify that Basic ElGamal encryption satisfied correctness (Definition 4).

3.3 Game hopping

We will write a sequence of games, where the first game (game 0) is the original challenger, and then make small modifications to the challenger each time. Adjacent games should be indistinguishable, but the last game should be impossible for the adversary to win.

Let S_i denote the event the adversary wins in game i. We want to bound $\Pr(S_0)$. We will seek to bound $|\Pr(S_i) - \Pr(S_{i+1})|$ and $\Pr(S_n)$.

Game hopping makes use of three types of operations:

1. **Transitions based on indistinguishability.** If the adversary can distinguish between game i and game $i+1$, then that gives a method for distinguishing two distributions assumed to be indistinguishable.
2. **Transitions based on failure events in both games.** Game i and game $i+1$ behave identically unless some failure event F occurs. Thus

$$S_i \wedge \neg F \Leftrightarrow S_{i+1} \wedge \neg F \ .$$

   The difference lemma then implies that

$$|\Pr(S_i) - \Pr(S_{i+1})| \leq \Pr(F) \ .$$

3. **Transitions based on failure events in one games.** Game i and game $i+1$ behave identically unless some failure event F occurs in game $i+1$. Thus

$$S_i \Leftrightarrow S_{i+1} \wedge \neg F \ .$$

   Conditional probability then implies

$$\Pr(S_i) = \Pr(S_{i+1} \wedge F) = \Pr(S_{i+1} \mid F) \Pr(F) \geq \Pr(S_{i+1}) \Pr(F) \ .$$

4. **Transitions based on rewriting.** Game $i+1$ is just a rewrite of game i, with only conceptual changes, but no mathematical changes.

3.4 Security of Basic ElGamal encryption

The semantic security of basic ElGamal encryption will be shown based on the security of the *Decisional Diffie–Hellman (DDH) problem* [Bon98].

Definition 5 (DDH problem)**.** *Let G be a cyclic group of prime order q and let g be a generator of G. Let $\mathcal{A}$ be an algorithm. The* decisional Diffie–Hellman (DDH) problem *for G is the task of distinguish triples (g^x, g^y, g^{xy}) ("real triples") from triples (g^x, g^y, g^z) ("random triples"), where $x, y, z \xleftarrow{\$} \mathbb{Z}_q$. Formally, define*

$$\mathbf{Adv}^{\mathsf{ddh}}_{G,g,q}(\mathcal{A}) = \left|\Pr\left(x, y \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(g^x, g^y, g^{xy}) = 1\right) - \Pr\left(x, y, z \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(g^x, g^y, g^z) = 1\right)\right| \ .$$

We will try to write down a proof of IND-CPA security of Basic ElGamal, then state the theorem after we've figured out what the proof proves.

Proof of semantic security of Basic ElGamal. Let $\mathcal{A}$ be an algorithm. The proof proceeds by a sequence of games.

Game 0. This is the original game.

Game 0 – IND-CPA with Basic ElGamal

$x \xleftarrow{\$} \mathbb{Z}_q$, $X \leftarrow g^x$
$(m_0, m_1, st) \xleftarrow{\$} \mathcal{A}(X)$
$b \xleftarrow{\$} \{0,1\}$
$y \xleftarrow{\$} \mathbb{Z}_q$, $c_1 \leftarrow g^y$, $Z \leftarrow X^y$, $c_2 \leftarrow m_b \cdot Z$
$b' \xleftarrow{\$} \mathcal{A}(st, (c_1, c_2))$
if $b = b'$ **then**
    return 1
else
    return 0
end if

Clearly,

$$\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{ElGamal}(G,g,q)}(\mathcal{A}) = \left| \Pr(S_0) - \frac{1}{2} \right| \ . \tag{1}$$

Game 1. This transition is based on indistinguishability. Game 1 is like game 0, except the value Z is computed as the power of a random exponent.

Game 1

$x \xleftarrow{\$} \mathbb{Z}_q$, $X \leftarrow g^x$
$(m_0, m_1, st) \xleftarrow{\$} \mathcal{A}(X)$
$b \xleftarrow{\$} \{0,1\}$
$y \xleftarrow{\$} \mathbb{Z}_q$, $c_1 \leftarrow g^y$, $\boxed{z \xleftarrow{\$} \mathbb{Z}_q,\ Z \leftarrow g^z}$, $c_2 \leftarrow m_b \cdot Z$
$b' \xleftarrow{\$} \mathcal{A}(st, (c_1, c_2))$
if $b = b'$ **then**
    return 1
else
    return 0
end if

Claim 1(a).

$$\Pr(S_1) = \frac{1}{2} \ . \tag{2}$$

Proof of Claim 1(a). In game 1, the value Z is uniformly random and independent of X and c_1, so it acts as a one-time pad of m_b. Thus, c_2 is also uniformly random and independent of X, c_1, and b. Thus, the adversary has no information about b. □

Claim 1(b).

$$|\Pr(S_0) - \Pr(S_1)| \leq \mathbf{Adv}^{\mathsf{ddh}}_{G,g,q}(\mathcal{B}^{\mathcal{A}}) \ , \tag{3}$$

where $\mathcal{B}$ is the following algorithm:

$\mathcal{B}^{\mathcal{A}}(U, V, W)$

$(m_0, m_1, st) \xleftarrow{\$} \mathcal{A}(U)$
$b \xleftarrow{\$} \{0,1\}$
$C \leftarrow m_b \cdot W$
$b' \xleftarrow{\$} \mathcal{A}(st, (V, C))$
if $b = b'$ **then**
    return 1
else
    return 0
end if

Proof of Claim 1(b). Algorithm $\mathcal{B}$ effectively "interpolates" between game 0 and game 1. If $\mathcal{B}$ receives a real triple (g^x, g^y, g^{xy}), then $\mathcal{B}^{\mathcal{A}}$ corresponds exactly to game 0, in which case

$$\Pr\left(x, y \xleftarrow{\$} \mathbb{Z}_q : \mathcal{B}^{\mathcal{A}}(g^x, g^y, g^{xy}) = 1\right) = \Pr(S_0) \ .$$

If $\mathcal{B}$ receives a random triple (g^x, g^y, g^z), then $\mathcal{B}^{\mathcal{A}}$ corresponds exactly to game 1, in which case

$$\Pr\left(x, y, z \xleftarrow{\$} \mathbb{Z}_q : \mathcal{B}^{\mathcal{A}}(g^x, g^y, g^z) = 1\right) = \Pr(S_1) \ .$$

By the definition of $\mathbf{Adv}^{\mathsf{ddh}}_{G,g,q}(\mathcal{B}^{\mathcal{A}})$, we have that

$$\mathbf{Adv}^{\mathsf{ddh}}_{G,g,q}(\mathcal{B}^{\mathcal{A}}) = |\Pr(S_0) - \Pr(S_1)| \ ,$$

which yields Claim 1(b). □

Combining equations (1), (3), and (2), we have

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{ElGamal}(G,g,q)}(\mathcal{A}) &= \left|\Pr(S_0) - \frac{1}{2}\right| && \text{(from equation (1))} \\
&\leq \left|\Pr(S_1) + \mathbf{Adv}^{\mathsf{ddh}}_{G,g,q}(\mathcal{B}^{\mathcal{A}}) - \frac{1}{2}\right| && \text{(from equation (3))} \\
&= \left|\frac{1}{2} + \mathbf{Adv}^{\mathsf{ddh}}_{G,g,q}(\mathcal{B}^{\mathcal{A}}) - \frac{1}{2}\right| && \text{(from equation (2))} \\
&= \mathbf{Adv}^{\mathsf{ddh}}_{G,g,q}(\mathcal{B}^{\mathcal{A}})
\end{aligned}$$

which yields the theorem below. □

Theorem 3 (Semantic security of Basic ElGamal encryption)**.** *Let G be a cyclic group of prime order q and let g be a generator of G. Let $\mathcal{A}$ be an algorithm, and let $\mathcal{B}$ be the algorithm above. Then*

$$\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{ElGamal}(G,g,q)}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{ddh}}_{G,g,q}(\mathcal{B}^{\mathcal{A}}) \ .$$

Game hopping is useful for complex schemes and protocols built of many cryptographic components because it allows us to swap them out one-at-a-time.

4 Digital signatures

(This section based in part on section 5 of David Pointcheval's chapter "Provable Security for Public Key Schemes" [Poi05].)

4.1 Basic definitions for digital signature schemes

Definition 6 (Digital signature scheme)**.** *For a message space $\mathcal{M}$, a* digital signature scheme *Σ is a triple of algorithms $\Sigma = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$.*

- $\mathsf{KeyGen}() \xrightarrow{\$} (sk, vk)$*: A probabilistic* key generation *algorithm that takes no input, and generates a (secret) signing key / (public) verification key pair.*
- $\mathsf{Sign}(sk, m) \xrightarrow{\$} \sigma$*: A probabilistic* signing *algorithm that takes as input a signing key sk and a message $m \in \mathcal{M}$, and outputs a signature σ.*
- $\mathsf{Ver}(vk, m, \sigma) \to \{0, 1\}$*: A deterministic* verification *algorithm that takes as input a verification key vk, a message m, and a signature σ, and outputs either* 0 *(invalid) or* 1 *(valid).*

$\underline{\mathbf{Exp}_{\Sigma}^{\mathsf{euf\text{-}cma}}(\mathcal{A})}$

1: // *Setup:*
2: $(sk, vk) \xleftarrow{\$} \mathsf{KeyGen}()$
3: // *Execution:*
4: $(m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sign}(sk,\cdot)}(pk)$
5: // *Winning condition:*
6: **if** $\mathsf{Ver}(m^*, \sigma^*, vk)$ and m^* was not asked to the Sign oracle **then**
7: **return** 1
8: **else**
9: **return** 0
10: **end if**

Figure 6: Security experiment for existential unforgeability under chosen message attack for digital signature schemes.

Definition 7 (Correctness)**.** *A digital signature scheme* Σ *is* correct *if for all* $m \in \mathcal{M}$*, all* $(sk, vk) \leftarrow \mathsf{KeyGen}()$*, and all* $\sigma \leftarrow \mathsf{Sign}(sk, m)$*, we have that* $\mathsf{Ver}(vk, m, \sigma) = 1$*.*

There are several possible security goals for the attacker:

- *Key recovery*: Compute sk, allowing the attacker to act as the signer.
- *Universal forgery*: For any message m, compute a valid signature for m.
- *Existential forgery*: Compute a valid signature σ for some m.

We can give the attacker different powers:

- *Key-only attack*: The attacker receives vk.
- *Known-message attack*: The attacker receives a list of message/signature pairs for a pre-selected list of messages.
- *Adaptive chosen-message attack*: The attacker may adaptively obtain signatures for messages of his choosing.

The "best" security notion is *existential unforgeability under adaptive chosen message attack (EUF-CMA)*. We want to bound

$$\mathbf{Succ}_{\Sigma}^{\mathsf{euf\text{-}cma}}(\mathcal{A}) = \Pr\left(\mathbf{Exp}_{\Sigma}^{\mathsf{euf\text{-}cma}}(\mathcal{A}) = 1\right) \ .$$

4.2 RSA Full-Domain Hash signature scheme

Let $H : \{0,1\}^* \to \mathbb{Z}_n^*$ be a hash function. The RSA Full-Domain Hash digital signature scheme [BR93], denoted $\mathsf{RSA\text{-}FDH}$, is shown in Figure 7 below.

4.3 The random oracle model

The proof will make use of the following intuition: for the adversary to be able to forge a signature for a message m, he should have to query H on that message.

In the *random oracle model* [BR93], we pretend H is a random function $\mathcal{H}$ as in Figure 8 that the challenger implements, so that the adversary can only obtain hash values by asking the challenger for them.

$\underline{\mathsf{KeyGen}()}$

1: Pick large random primes p and q
2: $n \leftarrow pq$
3: Pick $e \in \mathbb{Z}_n$ such that $\gcd(e, \phi(n)) = 1$
(recall $\phi(n) = (p-1)(q-1)$)
4: $d \leftarrow e^{-1} \mod \phi(n)$
5: **return** $(sk, vk) \leftarrow (d, (n, e))$

$\underline{\mathsf{Sign}(sk = d, m)}$

1: **return** $\sigma \leftarrow (H(m))^d \mod n$

$\underline{\mathsf{Ver}(pk = (n, e), m, \sigma)}$

1: $h \leftarrow \sigma^e \mod n$
2: **if** $h = H(m)$ **then**
3: **return** 1
4: **else**
5: **return** 0
6: **end if**

Figure 7: RSA Full-Domain Hash digital signature scheme algorithms.

$\underline{\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_n^*}$

1: // *Initialization*
2: $HList \leftarrow \emptyset$
3: // *Upon query* $\mathcal{H}(x)$
4: **if** $(x, z) \in HList$ for any z **then**
5: **return** z
6: **else**
7: $z \xleftarrow{\$} \mathbb{Z}_n^*$
8: Add (x, z) to $HList$
9: **return** z
10: **end if**

Figure 8: Random oracle $\mathcal{H}$

4.4 Security of RSA-FDH

The EUF-CMA security of RSA-FDH will be shown based on the difficulty of the *RSA problem*.

Definition 8 (RSA problem)**.** *Let $n = pq$ be the product of distinct primes of length λ and e be an integer relatively prime to $\phi(n)$. Given n, e, and $y \xleftarrow{\$} \mathbb{Z}_n^*$, compute x such that $x^e \equiv y \mod n$. If $\mathcal{A}$ is an algorithm, we denote $\mathcal{A}$'s probability of success as*

$$\mathbf{Succ}_{\lambda}^{\mathsf{rsa}}(\mathcal{A}) = \Pr\left(y \xleftarrow{\$} \mathbb{Z}_n^* : x \xleftarrow{\$} \mathcal{A}(n, e, y) : x^e \equiv y \mod n\right) .$$

Proof. Proof of EUF-CMA security of RSA Full-Domain Hash in the random oracle model. Let $\mathcal{A}$ be an algorithm. The proof proceeds by a sequence of games. We will be a little less detailed this time around.

Game 0. This is the original EUF-CMA game for RSA-FDH. Hence,

$$\mathbf{Succ}_{\mathsf{RSA\text{-}FDH}}^{\mathsf{euf\text{-}cma}}(\mathcal{A}) = \Pr(S_0) . \tag{4}$$

Game 1. In this game, we guess the index c of when m^* will be first asked to $\mathcal{H}$. Let $\mathsf{GoodGuess}$ be the event we guess correctly. If we guess failed, we abort the game. If $\mathsf{GoodGuess}$ occurs, then game 0 and game 1 behave identically, so we can apply a transition based on a failure event in one game. Assuming the adversary makes q_s queries to Sign and q_h queries to $\mathcal{H}$, there are $q_s + q_h + 1$ queries overall, so $\Pr(\mathsf{GoodGuess}) = \frac{1}{q_s + q_h + 1}$. Thus,

$$\Pr(S_1) \geq \Pr(S_0) \times \frac{1}{q_s + q_h + 1} . \tag{5}$$

Game 2. Here we will make use of the y from the RSA problem challenge. When answering the cth query to $\mathcal{H}$, return y instead.

Since y was chosen at random from $\mathbb{Z}_n^*$, the distributions are identical, so

$$\Pr(S_2) = \Pr(S_1) \ . \tag{6}$$

Game 3. Now we will make use of the n and e from the RSA problem challenge.

- When answering the cth query to $\mathcal{H}$, return y as in Game 2.
- For all other queries x to $\mathcal{H}$, choose a random $s \in \mathbb{Z}_n^*$, compute $z \leftarrow s^e \mod n$, and store (x, s, z) in $HList$. Return z.
- When asked to sign x, lookup (x, s, z) in $HList$ and return s.

Notice that signatures computed in this way are still valid, they are just computed differently by the challenger. Moreover, the distributions are identical, so

$$\Pr(S_3) = \Pr(S_2) \ . \tag{7}$$

Analysis of Game 3. If the attacker returns a forgery (m^*, σ^*), it must be that m^* was the cth query to $\mathcal{H}$, so $\mathcal{H}(m^*) = y$, and that $(\sigma^*)^e = \mathcal{H}(m^*) = y$. Thus, σ^* is a solution to the RSA problem for (n, e, y). Thus, game 3 acts as an algorithm $\mathcal{B}$ that, with the help of $\mathcal{A}$, solve the RSA problem.

$$\Pr(S_3) = \mathbf{Succ}_{\lambda}^{\mathsf{rsa}}(\mathcal{B}^{\mathcal{A}}) \ . \tag{8}$$

Combining equations (4)–(8) yields the result stated as the following theorem. □

Theorem 4 (EUF-CMA security of RSA-FDH)**.** *Let $\mathcal{A}$ be an algorithm, and let $\mathcal{B}$ be the algorithm given implicitly in game 3 of the above proof. Assume H is a random oracle, and that $\mathcal{A}$ makes q_h random oracle queries and q_s signing queries. Then*

$$\mathbf{Succ}_{\mathsf{RSA\text{-}FDH}}^{\mathsf{euf\text{-}cma}}(\mathcal{A}) \leq (q_s + q_h + 1)\, \mathbf{Succ}_{\lambda}^{\mathsf{rsa}}(\mathcal{B}^{\mathcal{A}}) \ .$$

This is an example of a *non-tight* reduction, because there is a factor of $(q_s + q_h + 1)$ between the two success probabilities.

5 Remark

It is important to realize that just because a scheme is "provably secure", it does not mean that it actually is secure. Koblitz and Menezes, in their series of papers on "Another Look at Provable Security" [KM04, KM06], take a critical view of the provable security paradigm. Sometimes proofs are wrong. Sometimes our security definitions do not capture the properties that are relevant in practice. Sometimes people instantiate schemes in practice with parameters that do not correspond with the probabilities and non-tightness of the security reduction. And of course security of cryptosystems can be undermined by poor implementations, attacks on humans, or simply circumventing the cryptography entirely. Nonetheless, security proofs offer an indication that at least some classes of attacks are ruled out, and allows cryptographers and cryptanalysts to focus their efforts on underlying hard problems.

References

[BDPR] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. URL http://www.cs.ucdavis.edu/~rogaway/papers/relations.pdf. Short version published as [BDPR98].

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology – Proc. CRYPTO '98*, *LNCS*, volume 1462, pp. 26–45. Springer, 1998. DOI:10.1007/BFb0055718. Full version available as [BDPR].

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In Joe P. Buhler, editor, *Proc. Algorithmic Number Theory 3rd International Symposium (ANTS) '98*, *LNCS*, volume 1423, pp. 48–63. Springer, 1998. DOI:10.1007/BFb0054851. EPRINT http://crypto.stanford.edu/~dabo/abstracts/DDH.html.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security (CCS)*, pp. 62–73. ACM, 1993. DOI:10.1145/168588.168596.

[Can01] Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols (extended abstract). In *Proc. 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS) 2001*, pp. 136–145. IEEE Press, 2001. DOI:10.1109/SFCS.2001.959888.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – Proc. CRYPTO '84*, *LNCS*, volume 196, pp. 10–18. Springer, 1984. DOI:10.1007/3-540-39568-7_2.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, **28**(2):270–299, 1984. DOI:10.1016/0022-0000(84)90070-9.

[KM04] Neal Koblitz and Alfred J. Menezes. Another look at "provable security", 2004. EPRINT http://eprint.iacr.org/2004/152. Published as [KM07].

[KM06] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". II, 2006. EPRINT http://eprint.iacr.org/2006/229.

[KM07] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, **20**(1):3–37, 2007. DOI:10.1007/s00145-005-0432-z. Earlier version appeared as [KM04].

[Mau05] Ueli Maurer. Abstract models of computation in cryptography. In Nigel P. Smart, editor, *Cryptography and Coding – 10th IMA International Conference*, *LNCS*, volume 3796, pp. 1–12. Springer, 2005. DOI:10.1007/11586821_1.

[Mau12] Ueli Maurer. Constructive cryptography — a new paradigm for security definitions and proofs. In Sebastian Mödersheim and Catuscia Palamidessi, editors, *Theory of Security and Applications*, *LNCS*, volume 6993, pp. 33–56. Springer, 2012. DOI:10.1007/978-3-642-27375-9_3.

[Poi05] David Pointcheval. *Advanced Course on Contemporary Cryptology*, chapter Contemporary Cryptology Provable Security for Public Key Schemes, pp. 133–189. Advanced Courses CRM Barcelona. Birkhuser, June 2005. DOI:10.1007/3-7643-7394-6_4.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, **28**(4):656–715, October 1949. URL http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology – Proc. EUROCRYPT '97*, *LNCS*, volume 1233, pp. 256–266. Springer, 1997. DOI:10.1007/3-540-69053-0_18.

[Sho06] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs, 2006. URL http://www.shoup.net/papers/games.pdf. First version appeared in 2004.