Lecture 6

Lecture date: Wed. 16,23 of February, 2005 Scribe: M. Bradonjić, S. Lu, J. Otchin

1 Digital Signatures

1.1 Introduction

Say Bob wants to communicate with Alice over a channel in which Eve can intercept and transmit messages. We have already considered the problem of message security - how Bob can encrypt a message for Alice such that Eve cannot decipher it in polynomial time with non-negligible probability.

Now we consider a different problem: if Bob receives a message purportedly from Alice, how can she be certain it isn't a forgery by Eve? One solution is to have Alice "sign" the message in some way that Bob will recognize, and that Eve will be unable to forge. There are a number of properties we would ideally like for such a signature:

- Alice can efficiently sign any message, for some reasonable limit on the message size.
- Given any document D that Alice has not signed, nobody can efficiently forge Alice's signature on D.
- Given a document D and a signature, anyone (not just Bob!) can efficiently tell whether the signature is valid for D.

We introduce digital signatures schemes as a way of accomplishing this.

1.2 Digital Signatures

We now give a definition of a digital signature scheme. A *digital signature scheme* is a triple of poly-time computable algorithms (KeyGen,Sign,Verify) over a message space $\mathcal{M}$ that satisfy the following conditions:

1. KeyGen$(1^n, R)$ is a probabilistic(with coin flips R) poly-time algorithm that outputs a public key and a secret key pair, (PK, SK)

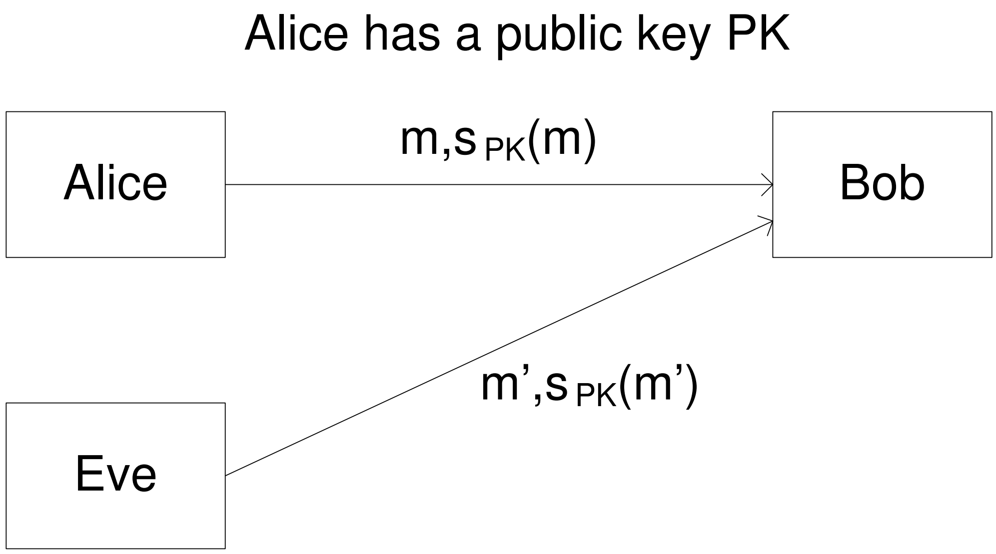

Figure 1: Eve tries to forge a signature.

2. $\mathsf{Sign}(D, PK, SK, R)$ is a probabilistic(with coin flips R) poly-time algorithm that signs a document $D \in \mathcal{M}$ with a signature $\sigma(D)$. Note: $|\sigma(D)|$ should be polynomially related to $|D|$

3. $\mathsf{Verify}(PK, D, s)$ is a (probabilistic) poly-time algorithm that outputs an element of $\{\mathsf{Yes},\mathsf{No}\}$. It returns Yes (with negligible error) if s is a valid signature of D, i.e. $s = \sigma(D)$

Given such a scheme, Alice can set up her document signer. First she generates $(PK, SK) \leftarrow \mathsf{KeyGen}(1^n, R)$ and publishes PK while keeping SK secret. Then when she wants to sign a document, D, she can run the signing algorithm $\sigma(D) \leftarrow \mathsf{Sign}(D, PK, SK, R)$ and sends the pair $(D, \sigma(D))$ to Bob. Bob can then verify the signature by running $\mathsf{Verify}(PK, D, \sigma(D))$. We also say that Eve forges a signature if she can produce a D and a $\sigma(D)$ (that was not signed by Alice) such that $\mathsf{Verify}(PK, D, \sigma(D)) = \mathsf{Yes}$ with non-negligible probability.

1.3 Security of a Digital Signature Scheme

When we talk about security for a digital signature scheme, we consider an adversary, Eve, who attempts to send a message to Bob and try to forge Alice's signature. What possible information does Eve have access to before attacking the system? Here are some reasonable assumptions that have been proposed:

- Eve only knows the public key. **(Key-only Attack)**
- Eve has seen a set of messages $\{m_1, \ldots, m_k\}$ with their corresponding signatures. The set of messages is given to her but not chosen by her. **(Known Message Attack)**
- Eve chooses a fixed set of messages $\{m_1, \ldots, m_k\}$ (there are two cases, where the messages are chosen independently of the public key or not) and gets to see the signatures of those messages. **(Chosen Message Attack)**

Once this information is given to her, what does it mean for the signature scheme to be broken by Eve?

- Eve computes the secret key. This is as bad as it gets because now Eve can sign *any* message she wants. **(Total Break)**
- Eve computes a poly-time algorithm that can forge a signature for any message. **(Universal Forgery)**
- Eve can forge a signature for a particular message of her choice. **(Selective Forgery)**

The previous examples will ultimately motivate our definition of security for signature schemes; but first we consider security after a single document is signed.

1.4 Lamport's 1-time signature scheme

The following is a construction of a one-time signature scheme (for messages of length n) out of a one-way permutation, $f : \{0,1\}^n \to \{0,1\}^n$.

1. $\mathsf{KeyGen}(1^n, R)$ will randomly select $2n$ elements from $\{0,1\}^n$. We will label them $x_1^0, x_1^1, x_2^0, x_2^1, \ldots, x_n^0, x_n^1$. Then we compute $y_i^b = f(x_i^b)$ for all $1 \leq i \leq n$ and $b \in \{0,1\}$. The secret key SK and the public key PK will be
$$SK = \begin{pmatrix} x_1^0 & x_2^0 & \cdots & x_n^0 \\ x_1^1 & x_2^1 & \cdots & x_n^1 \end{pmatrix} \quad PK = \begin{pmatrix} y_1^0 & y_2^0 & \cdots & y_n^0 \\ y_1^1 & y_2^1 & \cdots & y_n^1 \end{pmatrix}$$
2. $\mathsf{Sign}(m, PK, SK, R)$ will use $m = m_1 m_2 \ldots m_n$ to index SK meaning it will return $(x_1^{m_1}, x_2^{m_2}, \ldots, x_n^{m_n})$. For example if $m = 100\ldots1$ then we will return the selected entries:
$$\sigma(m) = \left(\begin{array}{ccccc} x_1^0 & \boxed{x_2^0} & \boxed{x_3^0} & \cdots & x_n^0 \\ \boxed{x_1^1} & x_2^1 & x_3^1 & \cdots & \boxed{x_n^1} \end{array} \right)$$

3. $\mathsf{Verify}(PK, m, s)$ where $m = m_1 m_2 \dots m_n$ and $s = (s_1, s_2, \dots, s_n)$ checks that $f(s_i) = y_i^{m_i}$ and returns Yes if they are equal for all $1 \le i \le n$. Continuing our previous example where $m = 100\dots1$ we check the equalities in the selected entries:

$$s = (s_1, s_2, \dots, s_n) \qquad \begin{pmatrix} y_1^0 & \boxed{f(s_2) = y_2^0} & \boxed{f(s_3) = y_3^0} & \cdots & y_n^0 \\ \boxed{f(s_1) = y_1^1} & y_2^1 & y_3^1 & \cdots & \boxed{f(s_n) = y_n^1} \end{pmatrix}$$

Assuming f is a one-way permutation, we claim that this scheme is secure against an adversary that is allowed only *one* query for the signature of a message m of his choice, then has to come up with $m' \neq m$ and a forgery $\sigma(m')$.

Proof We shall prove the contrapositive of the claim. We start by assuming there is an adversary A that can forge signatures with non-negligible probability, then we show that A can be used to invert f with non-negligible probability. Suppose A can forge a signature with probability$> \varepsilon$ conditioned over all m and PK (because f is a permutation, PK simply has a uniform distribution). Then we construct an algorithm to invert y as follows:

$(PK, SK) \leftarrow \mathsf{KeyGen}(1^n, R)$
$i' \leftarrow \{1, \dots, n\}; b' \leftarrow \{0, 1\}$
Replace $y_{i'}^{b'}$ in PK by y.
Give A this new PK, and he will request a signature for $m = m_1 m_2 \dots m_n$
if $m_{i'} = b'$ **then** FAIL; **else** send A the correct signature
A will then output a forged signature $(s_1, s_2, \dots, s_n)$ for a different message m'
if $m'_{i'} = b'$ **then** FAIL; **else** return $s_{i'}$

Notice y is uniformly distributed because f is a permutation, so the modified PKs look like they are also uniformly distributed, which means that the adversary will invert with the same probability$> \varepsilon$. The first place our algorithm can fail is if the adversary asks us to sign a message that has b' as its i'-th bit because we do not know $f^{-1}(y)$. This occurs with probability $\frac{1}{2}$. The second place our algorithm can fail is if the message generated by our adversary did *not* pick out b' as its i'-th bit, which means that we did not get the inverse to y from him. Because $m' \neq m$, it must differ by at least one bit, which means the chance that it differs on the i'-th bit is $\frac{1}{n}$.

Provided that our algorithm did not fail, then the answer it returns will be $x = s_{i'}$. Because s is a valid signature for m', it must be the case that $f(x) = f(s_{i'}) = y_{i'}^{m_{i'}} = y_{i'}^{b'} = y$. Thus the algorithm has succeeded in inverting y. Combining all the probabilities, we have a probability $\frac{1}{2} \cdot \frac{1}{n} \cdot \varepsilon$ of inverting f. Thus, f cannot be one-way, which proves the contrapositive. ■

Remark This scheme is only secure for signing one message, because the signature reveals part of your secret key. For example, if you signed $00\dots0$ and $11\dots1$, then your *entire* secret key has been revealed.

Some drawbacks of this scheme is that the public key has size $2n^2$ and that it can only be used to securely sign *one* message. In the next section we will construct a scheme that can sign many messages.

1.5 Security over multiple messages

The preceding section gave a signature scheme to sign any one message, after which forging a new signature is as hard as inverting a one-way function. However, it is clear that Lamport's algorithm is not at all secure if the adversary is allowed to see two distinct signed messages. This motivates the following definition of security over multiple messages(Goldwasser, Micali, Rivest):

Definition 1 *A digital signature scheme is* existentially unforgeable under an adaptive chosen-message attack *if for all $A \in PPT$ who is allowed to query* Sign *polynomially many times (such messages may be dependent on both previously chosen messages and the public key) cannot forge* any *new message.*

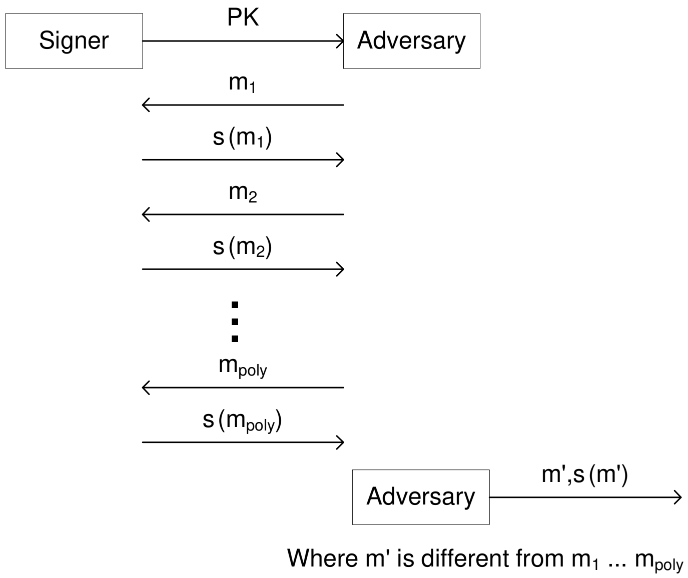

Figure 2: An Adaptive Chosen Message Attack

This will be chosen as our definition of security for a signature scheme. We mention that the definition is quite strong. For instance, consider the following scheme proposed by Diffie and Hellman: let $(\mathsf{Gen}, f, f^{-1})$ be a one-way permutation, where Gen outputs a (public) key k and a (private) trapdoor td. Given a document D, Sign gives the signature $\sigma(D)=f_k^{-1}(D)$. To verify a signature, any party can compute whether $f_k(\sigma) = D$. Therefore this indeed defines a legitimate signature scheme. But is it secure in the sense of the above definition?

It turns out that the scheme is not secure. An adversary can pick a random σ and define D to be $f_k(\sigma)$. Note that D could be completely meaningless as a document, but nevertheless the adversary has given a new document and a valid signature, so by definition, the scheme is not secure.

1.6 Signing multiple messages

We saw a secure scheme that could sign one message, and from this we can build a scheme that can sign many messages. All the schemes we will present in this section will require the signer to save a "state" based on how many messages have been already signed. Under the formal definition of a digital signature scheme this is not allowed, but we will relax this condition. Secure, stateless signature schemes have been first achieved by Goldreich in 1986.

One way we can sign N messages is to generate N secret key and public key pairs $\{(PK_0, SK_0), \ldots, (PK_N, SK_N)\}$ under the Lamport 1-time signature scheme. Then to sign the i-th message, one can simply sign it using (PK_i, SK_i) under the Lamport scheme. This way of signing multiple messages is highly impractical because our public key is unreasonably huge and we need to know a priori how many messages to sign.

By introducing hash functions into our schemes, we can make a great deal of improvement on the lengths of messages we can sign, and how many we can sign. For example, one may first hash a message using a collision resistant hash function and then sign the hashed document. Merkle in 1989 constructed a multi-time signature scheme using trees by "hashing-down" public keys to a single root which will be the master public key.

1.7 Introduction to Hash Functions

We want to define the notion of a hash function. Intuitively, it should be a function that is easy to compute, and the output should be shorter in length than the input.

Definition 2 *A hash function* $h : \{0,1\}^n \to \{0,1\}^m$ *is a deterministic, poly-time computable function such that* $n > m$*.*

Because h is length decreasing, the function is many-to-one, i.e. there exists a pair (x, x') such that $h(x) = h(x')$. In cryptographic constructions using a family of hash functions we want to have control over how these collisions occur. We consider a family of hash functions defined by a key-gen algorithm $\mathsf{Gen}(1^n, R) \to h$ where R is some randomness. We define the following three notions of collision resistance on the family, $\mathcal{H}$, of hash functions generated by $\mathsf{Gen}(1^n, R)$:

Definition 3 ***(Collision resistance)*** $(\forall A \in PPT)(\forall c > 0)(\exists N_c)(\forall n \geq N_c)$

$Pr_{H,R}[h \leftarrow H_n; (x_n, y_n) \leftarrow A(h) : h(x_n) = h(y_n)] \leq \frac{1}{n^c}$.

Definition 4 ***(Universal one-way [Naor-Yung])*** $(\forall A \in PPT)(\forall c > 0)(\exists N_c)(\forall n \geq N_c)$

$Pr_{H,R}[(x_n, \alpha) \leftarrow A(1^n); h \leftarrow H_n; y_n \leftarrow A(\alpha, h, x_n) : h(x_n) = h(y_n)] \leq \frac{1}{n^c}$.

Definition 5 ***(Two-wise independence)*** $(\forall A \in PPT)(\forall c > 0)(\exists N_c)(\forall n \geq N_c)$

$Pr_{H,R}[(x_n, y_n) \leftarrow A(1^n); h \leftarrow H_n : h(x_n) = h(y_n)] \leq \frac{1}{n^c}$.

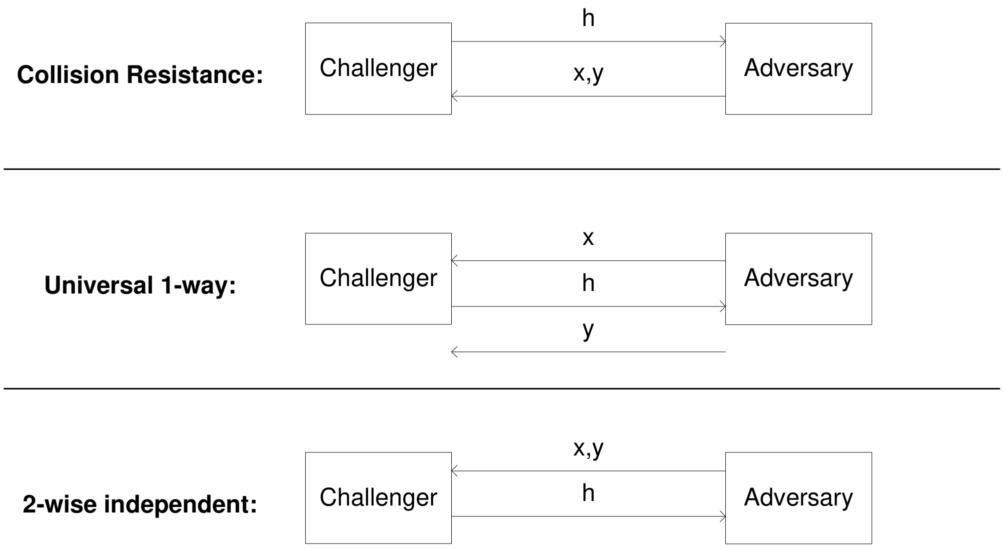

Figure 3: Three games of collision resistance.

1.8 Naor-Yung trees

We will present an improvement of Merkle's signature scheme based on the work of Naor-Yung. The construction assumes the one-time security of Lamport's 1-time signature, (KeyGen,Sign,Verify) as well as the existence of a family of universal 1-way hash functions $\{h : \{0,1\}^{4n^2} \rightarrow \{0,1\}^n\}$ which will be used in the construction of our signature scheme which signs $n^{log(n)}$ messages of length n. The main concept for signing is to first pretend we have a complete tree of height, say, $k = log^2(n)$ which will have an secret key and a public key at each node. Only the public key PK of the root of this tree will be published, which means our public key size is independent of the number of messages we need to sign. To sign the i-th message, we sign it on the i-th leaf using Lamport's 1-time signature, then include the public keys of the nodes in the path from the leaf to the root and their siblings. To make sure this path is authentic, we also need to have each parent sign the public keys of its two children. This is accomplished by concatenating the public keys of the two children then applying a hash to it, then signing the result. All of this information is to be included in the signature, but the good news is that the size of the signature does not grow as the number of signed messages increases. Notice that because we only pretended to have such a tree, some of these values may need to be computed and stored on the fly, but still this only takes poly-time to accomplish. Also notice that because our tree has more than polynomially many leaves, no polynomially bounded adversary can exhaust all of the leaves, so that we can always sign a message when an adversary asks for one.

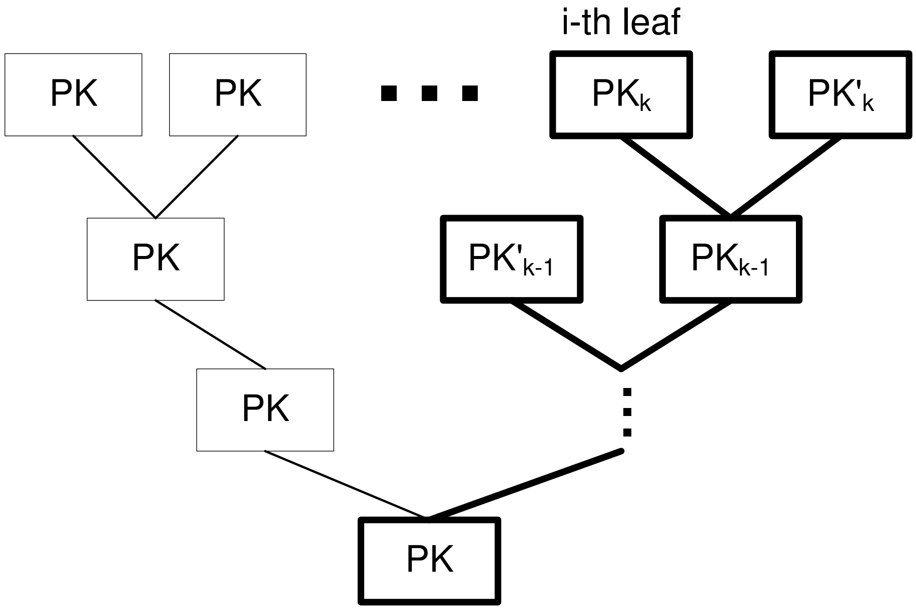

Figure 4: Signing the i-th message.

More formally, we can define a triple (MKeyGen,MSign,MVerify) with state information as follows:

- $\mathsf{MKeyGen}(1^n, R)$ will generate the keys for the root of our tree from the 1-time generator $(PK, SK) \leftarrow \mathsf{KeyGen}(1^n, R)$

- $\mathsf{MSign}(m, PK, SK, R)$ will use state information to sign a message. To sign the i-th message, m, we first set up some notation. Let n_k^j denote the j-th node (reading the tree from left to right) of depth $k = log^2(n)$. Then by this notation $n_{k-1}^{j/2}$ denotes its parent, $n_{k-2}^{j/4}$ denotes its grandparent, and so on. Let $n_k = n_k^i, n_{k-1} = n_{k-1}^{i/2}, \ldots$ and let $r = n_1$ denote the root of our tree. Let (PK_ℓ, SK_ℓ) be the keys corresponding to node n_ℓ for $1 \leq \ell \leq k$ with $(PK_1, SK_1) = (PK, SK)$. Also, let (PK'_ℓ, SK'_ℓ) denote the keys for the sibling of n_ℓ for $2 \leq \ell \leq k$. Finally, we compute $\sigma \leftarrow \mathsf{Sign}(m, PK_k, SK_k, R)$ and $\sigma_\ell \leftarrow \mathsf{Sign}(h(PK_{\ell+1} PK'_{\ell+1}), PK_\ell, SK_\ell, R)$ for $1 \leq \ell \leq k-1$, and output $(PK_\ell, PK'_\ell, \sigma_\ell, \sigma)$ as our signature for m.

- $\mathsf{MVerify}(PK, m, s)$ will first check that s is of the form $s = (PK_\ell, PK'_\ell, \sigma_\ell, \sigma)$. Then it will run $\mathsf{Verify}(PK_k, m, \sigma)$ and $\mathsf{Verify}(PK_\ell, h(PK_{\ell+1}PK'_{\ell+1}))$ and return Yes if they both pass.

We mention as a side note that the space requirements can be reduced to a constant if one uses pseudorandom functions to generate the public keys. Because pseudorandom outputs are indistinguishable from a uniform distribution, such a construction is equally secure.

This (stateful) digital signature scheme we constructed is existentially unforgeable adaptively secure if the Lamport 1-time scheme is secure and universal one-way hash functions exist. The sketch of the proof is as follows:

Sketch of Proof Assume for the contrapositive that there exists a poly-time adversary A that can forge a signature with non-negligible probability, ε after $p = poly(n)$ steps. Because there is a path of signatures from each leaf down to the root, two cases can occur in a forgery (1) A found a collision to h or (2) A can sign a message on an existing leaf or a message different from $h(PK_\ell PK'_\ell)$ on an existing node. One of the two cases occur with probability at least $\varepsilon/2$ so either we can show h is not universal 1-way, or we can show f is not one-way, which proves the contrapositive. ■

We summarize this result as showing secure digital signatures exist if one-way permutations exist (for the Lamport scheme to work), and universal 1-way hash functions exist. In the next section we will show how to construct a universal 1-way hash function from a one-way permutation.

1.9 One-way Permutations Imply Universal 1-way Hash Functions

We will construct a family of universal 1-way hash functions from a one-way permutation f. The hash functions we construct will take n bits to $n-1$ bits and they will be indexed by $h = (a, b)$ where $a, b \in GF(2^n)$. The algorithm for hashing a string x of length n is to apply $y \leftarrow f(x)$ then $z \leftarrow chop(ay+b)$ to $n-1$ bits (operations are taken over $GF(2^n)$). By the fact that f and the linear map $ay+b$ are both 1-1 and $chop$ is 2-1, our hash function is a 2-1 mapping from $\{0,1\}^n \to \{0,1\}^{n-1}$. We claim that this is a family of universal 1-way hash functions if f is a one-way permutation.

Proof Assume for the contrapositive that this family is not universal 1-way. Let A be a poly-time adversary that chooses x and is given h chosen from a uniform distribution can find a x' such that $h(x) = h(x')$ with probability$> \varepsilon$. Then to invert $y' = f(\gamma)$, we first look at x and we solve for (a, b) to satisfy the equation $chop(af(x)+b) = chop(af(\gamma)+b)$. Because f is a permutation, and the fact that this linear equation does not skew the distribution of the (a, b) returned, the hash function $h = (a, b)$ looks as if it were chosen truly from a uniform distribution. Then A will return x' such that $chop(af(x)+b) = chop(af(x')+b)$, but $f(x)$ and $f(\gamma)$ are the only two solutions to that linear equation, so $f(x') = f(\gamma) = y'$. Thus we can invert f with probability$> \varepsilon$, proving that it is not one-way. ■

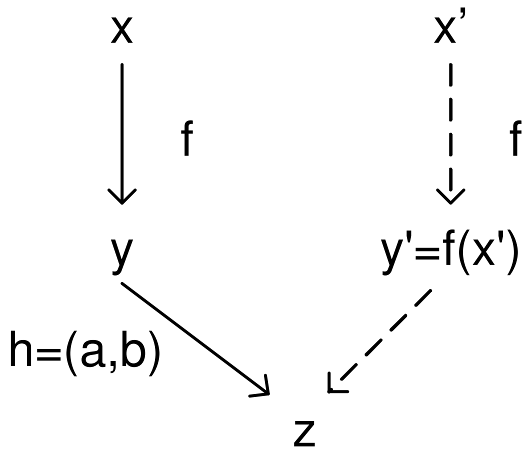

Figure 5: Setting a trap.

Assuming that there exists a family of one-way permutations: $f_{poly(n)}, f_{poly(n)-1}, \ldots, f_{n+1}$, such that $f_i : \{0,1\}^i \to \{0,1\}^i$, we may construct a family of universal 1-way hash functions $h : \{0,1\}^{poly(n)} \to \{0,1\}^n$ as follows:

$x_0 \leftarrow \{0,1\}^{poly(n)}$
for$k = 0$ **to**$poly(n) - n - 1$
  $a_k \leftarrow GF(2^{poly(n)-k}); b_k \leftarrow GF(2^{poly(n)-k})$
  $x_{k+1} \leftarrow chop(a_k f_{poly(n)-k}(x_k) + b_k)$
Output $x_{poly(n)-n}$

Proof Idea

Assume that there was a poly-time adversary A who could break this construction with probability$> \varepsilon$. Then we can pick a random location in our chain to set a trap by solving for (a_k, b_k) at that level as before. If the adversary finds a collision, then at some level in our construction there will be a collision. The probability of which that level will be equal to the one we set the trap is $\frac{1}{poly}$. This will allow for us to invert the one-way permutation at that location with probability$> \frac{\varepsilon}{poly}$, thus contradicting the one-way property of the function.

■

Thus we have shown the following theorem:

Theorem 6 *Hash functions exist if one-way permutations exist.*

1.10 Application to Signatures

Assuming that there exists a family of one-way permutations: $f_{4n^2}, f_{4n^2}, \ldots, f_{n+1}$, such that $f_i : \{0,1\}^i \rightarrow \{0,1\}^i$, we may construct a family of universal 1-way hash functions $h : \{0,1\}^{4n^2} \rightarrow \{0,1\}^n$ as seen in the previous section. This gives the following result:

Theorem 7 (Naor-Yung) *Secure digital signatures exist if one-way permutations exist.*

To conclude on this topic, we mention that it has also been shown:

Theorem 8 (Rompel) *Secure digital signatures exist iff any one-way function exists.*

1.11 Hash functions for Commitment Schemes

Consider the scheme where a sender is given a hash function $h : \{0,1\}^{2n} \rightarrow \{0,1\}^n$ by a receiver. Let h be applied on every $2n$ bits x, and then iteratively on every $2n$ bits of the result producing the final output a.

Claim 9 *The output a is a good commitment for x.*

Sketch of Proof The receiver gets no information on x which follows from the way we constructed our hash function. On the other hand, the sender cannot cheat because in order to cheat, he would be able to find a collision for the hash function h. ■