Notes for Introduction to Modern Cryptography (LMU, WS 2020/21)

Lecturers:
Sophia Grundner-Culemann, Dieter Kranzlmüller,
Tobias Guggemos and Ulrich Rührmair

Typing:
Yilun Yang

Contents

Part I

Introduction to Standard Digital Cryptography

Chapter 1

Symmetric cryptography

1.1 Basic setting

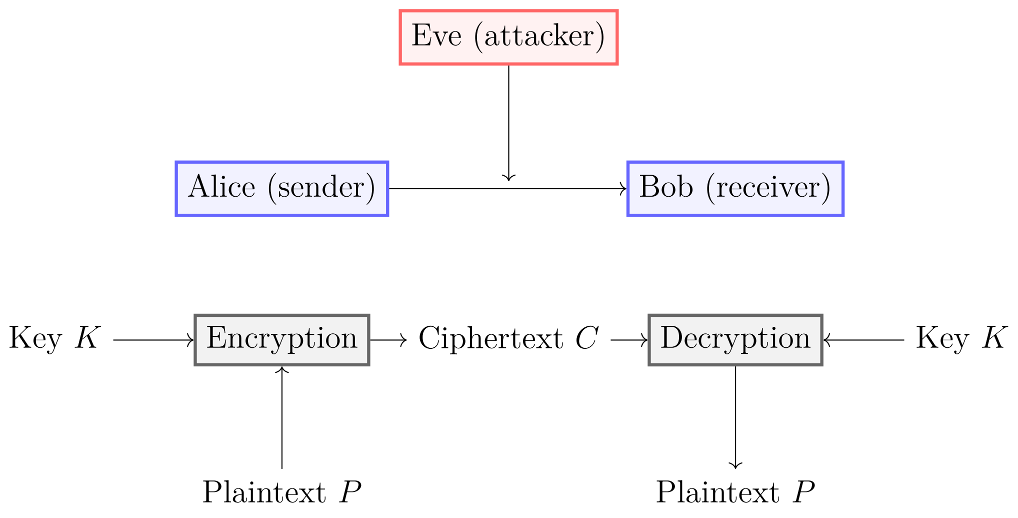


Figure 1.1: Symmetric encryption.

Symmetric encryption

- Alice and Bob share a same secret key K (*symmetric*).
- Eve is assumed to know the general method of encryption, decryption and choosing keys, and also the ciphertext C. Modern symmetric cryptographic algorithms apply the *Kerckhoffs' principle*: a crypto system should be secure even if everything about the system except the key is public knowledge.

- Eve does not know anything about the key K or the plaintext P, which are the attack targets.

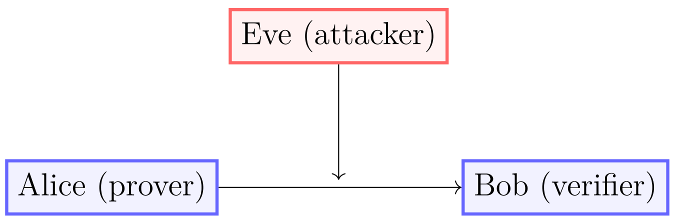
Definition 1.1. *A* ***cipher or symmetric encryption scheme*** *consists of three sets:*

- *A plaintext set $\mathcal{P}$ together with a probability distribution $D_{\mathcal{P}}$ on $\mathcal{P}$;*
- *A key set $\mathcal{K}$ together with a probability distribution $D_{\mathcal{K}}$ on $\mathcal{K}$;*
- *A ciphertext set $\mathcal{C}$.*

and two functions:

- *An encryption function* $\mathrm{Enc} : \mathcal{P} \times \mathcal{K} \to \mathcal{C}$, $(P, K) \mapsto \mathrm{Enc}_K(P)$*;*
- *A decryption function* $\mathrm{Dec} : \mathcal{C} \times \mathcal{K} \to \mathcal{P}$, $(C, K) \mapsto \mathrm{Dec}_K(C)$*.*

Abd decryption inverts encryption: $\mathrm{Dec}_K \circ \mathrm{Enc}_K = \mathbb{1}$, $\forall K \in \mathcal{K}$.



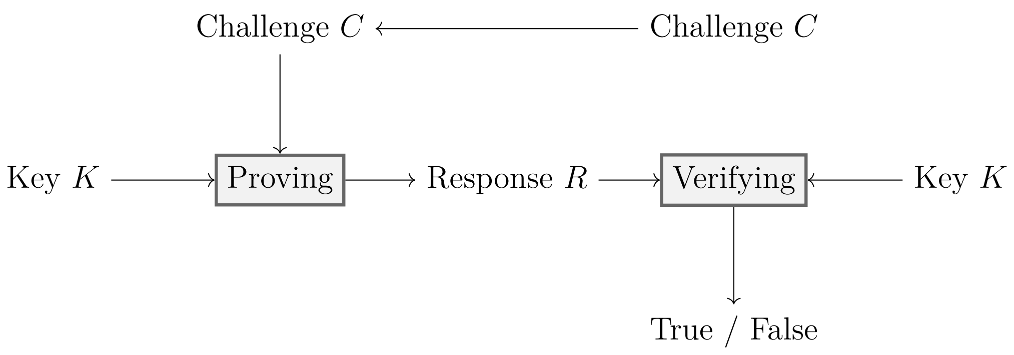


Figure 1.2: Symmetric identification.

Symmetric identification Now eve is assumed to:

- know general method of proving, verifying and of choosing keys;

- know previous challenge/response pairs;
- be able to actively interact with the prover.

The attack target is anything about Key K such that the verifying method returns True. The pros of using a symmetric encryption scheme is that the security of those is well understood. However, it requires a pre-shared key.

1.2 First attempts

Cesar / shift cipher Encryption method: shift every letter in alphabet by a constant number k of letters. However, the key space *only* has length $\log_2 26 \approx 5$ bits, and hence *direct attack* works.

Substitution cipher Use a substitution alphabet instead.

- Key length is now increased to $\log_2 26! \approx 89$ bits.
- A lot of structures of the plaintext is however still preserved. It can be attacked by *frequency analysis* (Al Kindi, 801-873).
- Even if plaintext language is unknown, the mere frequency distribution of symbols in ciphertext can often identify the used language.

Vigenere Cipher

- The key is a short code word K of length n. Plaintext letter at position $mn + i$ $(m, i \in \mathbb{Z})$ is applied a shift K_i.
- Attack (Friedrich W. Kasiski, 1805-1881): if the key is repeated over a (long) plaintext, occasionally the same word or phrase will be encrypted with the same letters of key, leading to *repetitions* in ciphertext. Then the key length is likely a divider of the distance of two repetitions (can be statistically confirmed).
- Once key length is known, we can regroup the ciphertext into n groups and do a frequency attack to each.

Intermediate Summary

- Described attacks required increasing amounts of ciphertext material, but all still were highly efficient.
- The options are

- 1 making the encryption method more complicated while keeping the key short $\Rightarrow$ modern, efficient block /stream ciphers with *non-provable security*, like DES and AES;

- 2 making the key longer while keep encryption method simple $\Rightarrow$ *provably secure*, but slightly impractical ciphers like the one-time pad.

1.3 One-time pad

The basic idea of one-time pad is to choose the key independently at random as long as the complete plaintext in the Vigenere cipher, and that any key cannot be reused for other plaintext later. In a equivalent binary version, it is defined as below:

Definition 1.2. ***(One-time pad as a cipher)*** *we call the following cipher the one-time pad (of length n):*

- *the plaintext set $\mathcal{P}$, key set $\mathcal{K}$ and ciphertext set $\mathcal{C}$ are all equal to $\{0, 1\}^n$;*

- $\mathrm{Enc}_K(P) = P \oplus K$, *where $\oplus$ is bitwise XOR function;*

- $\mathrm{Dec}_K(C) = C \oplus K$.

Given any (plaintext, ciphertext)-pair, there is exactly one key that maps this plaintext to this ciphertext. Therefore if the keys are uniformly distributed, the one-time pad is perfectly secure. To be more concrete,

Definition 1.3. *A cipher is called* ***perfectly secret (or information-theoretically secure)*** *if $\forall P \in \mathcal{P}, C \in \mathcal{C}$, the event that P occurs and C occurs are independent, namely*

$$\Pr(P|C) = \Pr(P), \ \forall \Pr(C) > 0. \tag{1.1}$$

Theorem 1.4. ***(Shannon, 1949)*** *A cipher is perfectly secret (or information-theoretically secure) if the probability distribution $D_{\mathcal{K}}$ on $\mathcal{K}$ is a uniform distribution, and for every plaintext $P \in \mathcal{P}$ and every ciphertext $C \in \mathcal{C}$, there is exactly one key $K_{P,C} \in \mathcal{K}$ such that*

$$\mathrm{Enc}_{K_{P,C}}(P) = C. \tag{1.2}$$

Proof.

$$\begin{aligned}\Pr(P|C) &= \frac{\Pr(P)\Pr(C|P)}{\Pr(C)} \\ &= \frac{\Pr(P)\Pr(K_{P,C})}{\sum_{P^*\in\mathcal{P}}\Pr(P^*)\Pr(K_{P^*,C})} \\ &= \frac{\Pr(P)}{\sum_{P^*\in\mathcal{P}}\Pr(P^*)} \\ &= \Pr(P).\end{aligned} \tag{1.3}$$

□

Corollary 1.5. *The one-time pad is a perfectly secret or information-theoretically secure encryption scheme.*

Theorem 1.6. ***(Shannon, 1949, semi-formal)*** *In order to encrypt arbitrary messages of length n bits in a perfectly secret or information-theoretically secure manner, uniformly chosen keys of length n bits must be employed, i.e., one must use n* ***true*** *and independent random bits in the choice of the keys and in the encryption process.*

Practical Impact

- Since ciphertext and plaintext are information-theoretically and statistically independent, even unbounded computational power does not help the attacker.
- The only assumptions required for perfect and everlasting secrecy are *uniform randomness* and *secrecy of the key itself.*
- However, the former one can be very cumbersome for large n in practice, while the latter is also non-trivial.
- More importantly, the one-time pads cannot be reused, as

  $$(P_1 \oplus K) \oplus (P_2 \oplus K) = P_1 \oplus P_2. \tag{1.4}$$

  The key cancels and some frequency analysis can be performed.

1.4 Stream and block ciphers

Perfect secrecy is a worthwhile goal, but its costs may be unnecessarily high as unbounded computational power does not exist. For practicability, a scheme may still be considered secure if it leaks only a tiny amount of information to eavesdroppers with bounded computational power. We relax the notion of perfect secrecy by the following:

- security is only preserved against efficient adversaries.
- adversaries can potentially succeed with some very small probability, which is small enough so that we are not concerned that it will ever really happen.

The adversaries $\mathcal{A}$ include

- brute force: exhaustive search of entire key space;
- attacking cipher text: with access to a set of valid ciphertexts, to find key(s) and / or generate clear text message;
- known plain text: with access to a set of clear plaintext-ciphertexts pairs encrypted with same key, to find key(s)and / or generate clear text message;
- chosen plain text: with access to service that encrypts texts of $\mathcal{A}$'s choice;
- chosen cipher text: with access to service that decrypts given ciphertexts.

1.4.1 Modern block ciphers

A single, short key (typically 64 to 256 bits) encrypts arbitrarily large amount of plaintext data (e.g., many GBs). The plaintext is typically grouped into blocks of 64 or 128 bits (eventually filled up with padding) and the ciphertext will also be a multiple of blocks. Prominent examples are

- data encryption standard (DES) - *considered broken*;
- 3DES - *should not be used*;
- advanced encryption standard (AES) - *current state of art.*

Data encryption standard (DES) Details are shown in Algorithm. 1. S-Boxes and Permutation lead to change the ciphertext by around 50% when changing a single plaintext bit, which means strong avalanche effect. The design of DES has shown no fundamental holes or gaps until today, but the key length is not long enough such that all possible keys can be tested by brute force:

- (”Deep Crack”, Electronic Frontier Foundation, 1998): Tests 88 billion keys / sec. Breaks DES in $\approx$ 56 hours. Costs \$250,000.
- (”COPACOBANA”, U Bochum / U Kiel, 2006): Tests 65 billionn keys / sec. Breaks DES in $\approx$ 6.4 days. Costs \$10,000.
- (”COPACOBANA”, 2008) Breaks DES in < 1 day.

To overcome the issues of DES, people designed

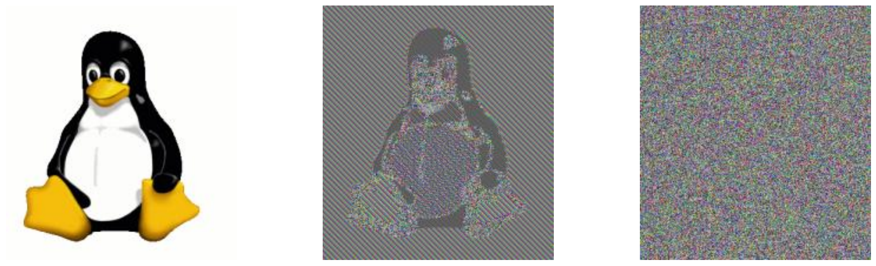
- Double-DES: $\text{DES}(\text{DES}(M, K_1), K_2)$. It is meant to double the complexity by doubling the key-size with expected key security level of 2^{2n}, while known plaintext attack weakens security to 2^{n+1}.
- Triple-DES: $\text{DES}(\text{DES}^{-1}(DES(m, K_1), K_2), K_3)$. The expected effective key size was $56 * 3 = 168$ bits, while it is actually $56 * 2 = 112$ bits because of ”meet in the middle attack”. 3DES is very inefficient even though considered secure now.

Advanced encryption standard (AES) Details are shown in Algorithm. 2 The Rijndael’s S-Box is designed to be resistant to linear and differential cryptanalysis. It is generated by determining multiplicative inverse for given number in the Galois field:

$$\text{GF}(2^8) = \text{GF}(2)/(x^8 + x^4 + x^3 + x + 1). \tag{1.5}$$

Security of AES

- Design (criteria) of AES had to be published.
- 50% of output bits depend of every input bit after 2 rounds.
- The algebraic construction of S-Box is also published; it is non-linear to a high degree.
- ShiftRow is introduced to hinder truncated differentials and square attack.



(a) original (b) ECB mode (c) Chain mode

Figure 1.3: Encryption of ”Tux” in ECB mode and in another encryption mode with block chaining.

- MixColumn grants high diffusion: change of one input byte causes potential changes in every output byte!
- When using 10 rounds for AES-128, brute force attacks are known for up to 7 rounds (“3 rounds spare” that can be raised comparatively easy).
- Exhaustive key search is hopeless. With the same key test rate as the DES-breaker ”Deep Crack”, an AES-attack by exhaustive key search would take $\approx$ 710 000 trillion years.

1.4.2 Encryption modes and attacks

Electronic Code Book (ECB) mode Every block of plaintext is encrypted with the same key. The problem is that, identical plaintext blocks result in identical ciphertext blocks, see Fig.1.3. To avoid this, we introduce chain mode encryption that is dependent on the ciphertext of previous block.

Cipher Block Chaining (CBC) mode In CBC mode, every plaintext block is firstly XORed with the previous ciphertext block, while for the first plaintext block, it requires an initialization vector (IV). The downsides of CBC are however

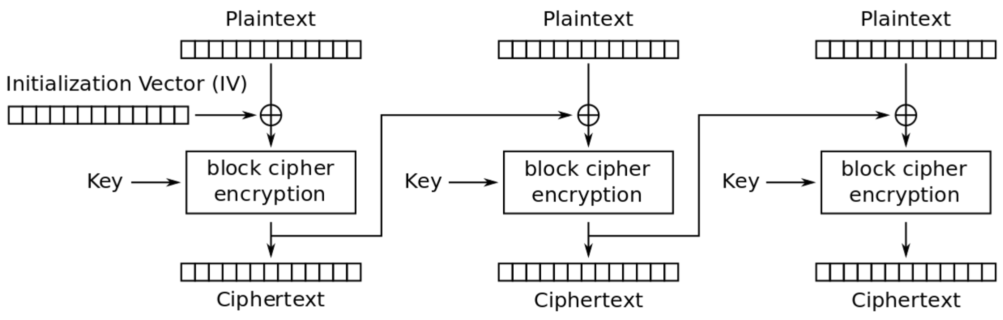
- Each block depends on the en-/decryption of the previous blocks so that *no* parallelization is possible.
- If IV is predictable, that allows *known-plaintext* attacks.

Algorithm 1 Data encryption standard

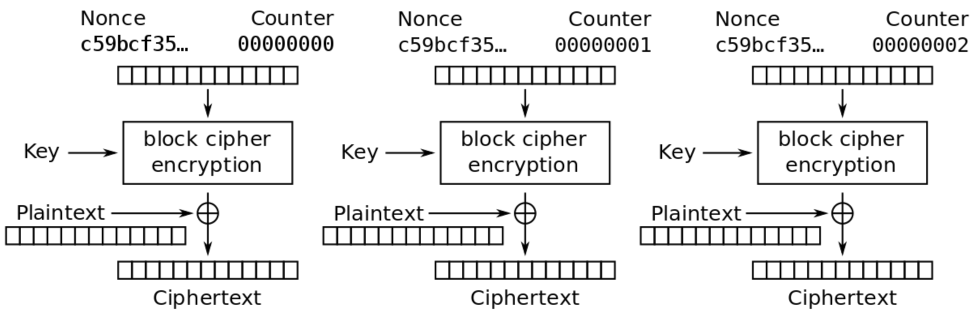
```
function RoundKey(64-bit Key $K_0$)
    $K' \leftarrow \sigma_{PC-1}(K_0)$ ($\sigma_{PC-1} \sim$ fixed permutation using 56 bits)
    $C_0 \leftarrow K'[0:28]$, $D_0 \leftarrow K'[28:56]$
    for $r \leftarrow 1, 16$ do
        $k_r \leftarrow 1$ or 2
        $C_r \leftarrow s_{k_r}(C_{r-1})$, $D_r \leftarrow s_{k_r}(D_{r-1})$
        ($s_i \sim$ left shift rotation of length $i$)
        $K_r \leftarrow \sigma_{PC-2}(C_r D_r)$
        ($\sigma_{PC-2} \sim$ fixed permutation using 48 bits)
    end for
    Return $K_r$
end function

function F(32-bit $R$, 48-bit $K$)
    $U \leftarrow R_E \oplus K$
    for $i \leftarrow 1, 8$ do
        $R_i \mapsto R[4(i-1)-1 : 4i+1]$
        $S_i \leftarrow S(R_i \oplus K[6(i-1), 6i])$
        ($S \sim$ S-Box maps 6 bits to 4 bits. Crucial for security!)
    end for
    Return $S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8$
end function

function DES(64-bit Message $M_0$, 48-bit Round Keys $K_r$)
    $M' \leftarrow \sigma_{IP}(M_0)$ ($\sigma_{IP} \sim$ 64 bits permutation)
    $L_0 \leftarrow M'[0:32]$, $R_0 \leftarrow M'[0:32]$
    for $r \leftarrow 1, 16$ do
        $L_r \leftarrow R_{r-1}$
        $R_r \leftarrow L_{r-1} \oplus F(R_{r-1}, K_r)$
    end forReturn $\sigma_{IP}^{-1}(L_{16} R_{16})$
end function
```

Algorithm 2 Advanced encryption standard

```
$N_b, N_k \leftarrow 4$, 6 or 8 (Block / key length: $N_{b,k} \times 4$ bytes)
$N_r = \max(N_b, N_k + 6)$ (Rounds)

function RoundKey($4N_k$ bytes Key $K_0$)
    for $r \leftarrow 1, N_r$ do
        $K_r[0,:] \leftarrow$S-Box$(s_1(K_{r-1}[N_k - 1, :])) \oplus K_{r-1}[0,:]$
        for $k \leftarrow 1, N_k - 1$ do
            if $N_k = 8$ and $k = 4$ then
                $K_r[k,:] \leftarrow$S-Box$(K_r[k-1,:]) \oplus K_{r-1}[k,:]$
            else
                $K_r[k,:] \leftarrow K_r[k-1,:] \oplus K_{r-1}[k,:]$
            end if
        end for
    end for
    Return $K_r$
end function

function MixRows(4 bytes $M$)
    $m(x) \leftarrow M[0] + M[1]x + M[2]x^2 + M[3]x^3$ $(M[i] \in \mathrm{GF}(2^8))$
    $s(x) \leftarrow m(x)(2 + x + x^2 + 3x^3) \mod x^4 + 1$
    Return $S$ s.t. $s(x) = S[0] + S[1]x + S[2]x^2 + S[3]x^3$
end function

function AES(Message $M_0$, Key $K_0$, Round Keys $K_r$)
    $C_0^K \leftarrow M_0 \oplus K_0$                                        (AddRoundKey)
    for $r \leftarrow 1, N_r$ do
        $C_r^{Sub} \leftarrow$ SubWord$(C_{r-1}^K)$, $C_r^{Shift}[:,0] \leftarrow C_r^{Sub}[:,0]$          (SubBytes)
        for $i \leftarrow 1, 3$ do
            $C_r^{Shift}[:,i] \leftarrow s_1(C_r^{Sub}[:,i])$
        end for                                                         (ShiftRows)
        if $r < N_r$ then
            for $j \leftarrow 0, N_b - 1$ do
                $C_r^{Mix}[j,:] \leftarrow$ MixRows $(C_r^{Shift}[j,:])$
            end for                                                     (MixRows)
        end if
        $C_r^K \leftarrow C_r^{Mix} \oplus K_r$                                    (AddRoundKey)
    end for
    Return $C_{N_r}^K$
end function
```



Cipher Block Chaining (CBC) mode encryption



Counter (CTR) mode encryption

Counter (CTR) mode The initialization vector consists of a random number (nonce, can be sent in plaintext) to be selected for each cipher, linked (via like XOR or concatenation) to a counter that is incremented with each additional block. The encryption and decryption are identical in CTR mode. CTR mode works similarly to one-time pads, where the IV should *not be reused*.

1.4.3 ChaCha20

As a modification of Salsa20 (205), ChaCha20 (2008) is a modern stream cipher. Due to its good software performance, it is nowadays often used for networking protocols, e.g. TLS, IPsec and QUIC. Like AES-CTR, it runs in different rounds (ChaCha20 means 20 rounds), acts on the state array (16 32-bit words for ChaCha20) and XORs the output with plaintext. However, instead of S-Box, the core function is ”Quarter Round” which is repeated

Algorithm 3 ChaCha20

```
function QuarterRound(32-bit words $\&a, \&b, \&c, \&d$)
    $a \leftarrow a + b \mod 2^{32}$
    $d \leftarrow a + d \mod 2^{32}$
    $d \leftarrow s_{16}(d)$
    $c \leftarrow c + d \mod 2^{32}$
    $b \leftarrow b + c \mod 2^{32}$
    $b \leftarrow s_{12}(b)$
    $a \leftarrow a + b \mod 2^{32}$
    $d \leftarrow a + d \mod 2^{32}$
    $d \leftarrow s_8(d)$
    $c \leftarrow c + d \mod 2^{32}$
    $b \leftarrow b + c \mod 2^{32}$
    $b \leftarrow s_7(b)$
end function

function DoubleRound(State $\&S$, Round i)
    if $i\%2 = 0$ then
        QuarterRound($S[0], S[5], S[10], S[15]$)
        QuarterRound($S[1], S[6], S[11], S[12]$)
        QuarterRound($S[2], S[7], S[8], S[13]$)
        QuarterRound($S[3], S[4], S[9], S[14]$)
    else
        QuarterRound($S[0], S[4], S[8], S[12]$)
        QuarterRound($S[1], S[5], S[9], S[13]$)
        QuarterRound($S[2], S[6], S[10], S[14]$)
        QuarterRound($S[3], S[7], S[11], S[15]$)
    end if
end function

function GetBlockCipher(Key $K$, CTR $c$, IV $v$)
    State $S$ (Initial State)
    $S[0:4] \leftarrow$ "expand 32-byte k"(constant)
    $S[4:12] \leftarrow k$
    $S[12:] \leftarrow c$
    $S[13:16] \leftarrow v$
    for $r \leftarrow 0, 19$ do DoubleRound($S, r$)
    end for
    Return $S$
end function
```

round times. The details are in Algorithm. 3.

Security of ChaCha20

- ChaCha20 is simplistic and the design is open;
- There is high diffusion after only 6 rounds;
- It has easy constant-time implementation;
- It is very resistant against timing attacks;
- Counter re-usage opens the same attack vector as in AES-CTR.

1.5 Hash functions

The motivation of introducing Hash functions is to verifying input of any length with some number. Thus it should be hard to reverse or to find "collisions". We will now go through a bunch of definitions.

Definition 1.7. *A **Hash function** h is a mapping*

$$h : \{0,1\}^* \to \{0,1\}^n, \tag{1.6}$$

where $n \in \mathbb{N}$ and $\{0,1\}^*$ is the set of arbitrary-length finite bitstrings.

By definition, it is *not injective.*

Definition 1.8. *A pair (x, y) such that $x \neq y$ and $h(x) = h(y)$ is called a **collision** of h.*

Definition 1.9. *Let $h : X \to Y$ and let $y \in Y$ be randomly picked. If it is hard to find an element of $h^{-1}(y)$, then h is called a **one-way function**.*

Note that there is *no proof* that one-way functions exist, but there are functions we currently think are "one-way".

Definition 1.10. *Let $h : X \to Y$ and let $x \in X$ be randomly picked. If it is hard to find $x' \in h^{-1}(h(x))$ such that $x' \neq x$, then h is **second-preimage resistant**.*

Definition 1.11. *Let $h : X \to Y$. If it is hard to find any collision of h, then h is **collision-resistant**.*

Apparently collision-resistance $\Rightarrow$ second-preimage resistance $\Rightarrow$ one-wayness.

Definition 1.12. *If a hash function h is collision-resistant, than it is called a **strong cryptographic hash function**. If h is second-preimage resistant, then it is called a **weak cryptographic hash function**.*

Birthday paradox The definition of cryptographic hash functions (we refer to the strong one if not specified) looks nice, but it directly comes with the "birthday paradox":

Theorem 1.13. ***(Birthday paradox)*** *We consider random choices, with replacement, among k elements. The expected number of choices until a collision occurs is $O(\sqrt{k})$.*

That means, given a k-bit Hash function that Alice produced, Eve can find a collision with probability $\geq 1/2$ by producing $\sqrt{2^k} = 2^{k/2}$ hashes. However depending on the hashing algorithm, this could still take enough time be worthless for an attacker.

Compression functions Hash functions can be built by using compression function H that maps $\{0,1\}^{n+k}$ to $\{0,1\}^n$. The input P can be divided into k-bit pieces P_i, while the hash function is initialized with n-bit IV C_0. Then by a sequence $C_i = H(C_{i-1}||P_i)$ ($||$ denote concatenation), we finally get the hash of length n.

- MD5 (Ron Rivest, 1991): $n = 128$, $k = 512$ with compression function H_{MD5}. It Used to be a standard in network security for many years, while first attacks against MD-family were in the 90s. Nail-in-the-coffin-attack is published in 2006/07 (Stevens, Lenstra, de Weger), which shows MD5 is *not secure*!
- SHA family:
  - SHA-1: it is insecure now.
  - SHA-2: also known as SHA256 or SHA124. It is still in use and considered secure.
  - SHA-3 (Keccak): standardized in 10/2012 by NIST as an alternative to SHA-2. Up to now it is considered secure; but still relatively young. (See Fig. 1.4.)

Message Authentication Code (MAC) MAC is designed to achieve *authentication* (verifying the source of message) and *integrity* (verifying that message was not manipulated). The idea is to calculate cryptographic hash of a message M where the hashing algorithm A depends on a (shared symmetric) key K. Therefore it is also called a *keyed hash function*.
For instance *CBC-MAC* which uses the result of the last block of AES-CBC is a MAC. Encryption is however not sufficient; eve can just manipulate the

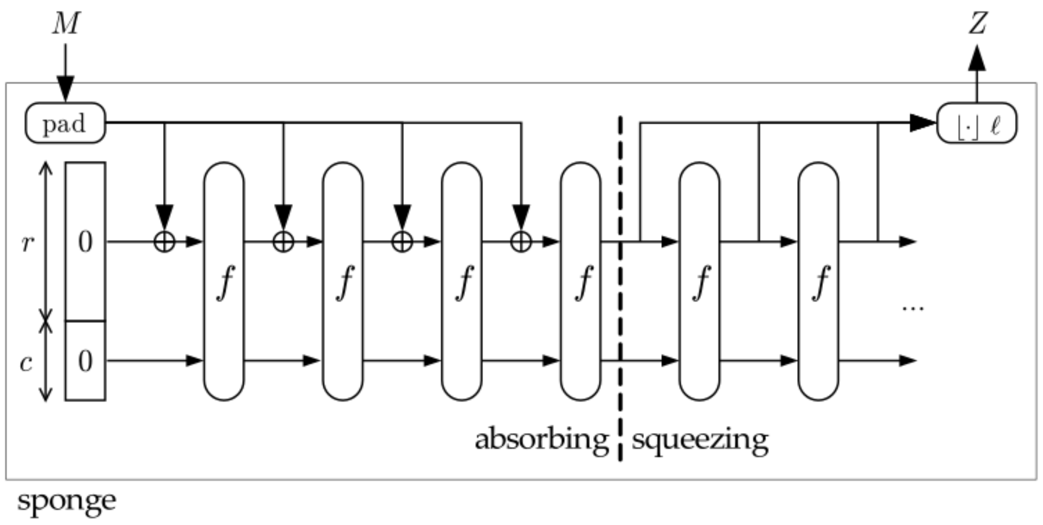


Figure 1.4: SHA-3 scheme

encrypted message without knowing anything about the plaintext! Thus the integrity is not achieved.
To solve this, one needs the *Hashed Message Authentication Code (HMAC).* Iy is not based on block-cipher, but on unkeyed cryptographic hash-function. As an example, we can take

- Merkle-Damgard-hash function $H : \{0,1\}^{2n} \to \{0,1\}^n$.
- two constants *opad*, *ipad* of length n.
- key k.
- $\text{HMAC}_k(m) = H(k \oplus opad||H(k \oplus ipad||m))$.

The outer hash is necessary because with Merkle-Damgard-hash functions, as $H(k||m)$ would be vulnerable to length extension attacks, i.e., Eve could compute a hash $H(k||m||pad||\text{extension})$ without knowing the key. HMAC is an industry standard and *widely used*, because it is efficient, easy to implement (if you know what you‘re doing) and has been proven secure under certain conditions.
CBC-MAC and HMAC are computationally secure. It is also possible to construct information-theoretically secure MACs where the *key length* is independent from the message size and the *probability* of a collision is (abstractly) known. We will get back here later, but not we just mention two important aspects:

- Their security is not threatened by growing computation power.

- To protect a message with a one-time-pad key, it is *not necessary* to generate a key as long as the message (which is also true for computationally secure MACs, but not obvious here).

Other uses of hash functions

- Pseudo-random number generators.
- Password-tables: Instead of storing passwords as plaintexts, websites hash the chosen password and store only the hash. With high probability, the password is correct if the hash is correct.
- Asymmetric cryptography: Authentication and integrity protection, e.g., through digital signatures.

Chapter 2

Asymmetric cryptography

2.1 Preparations

This section is aimed to prepare the stage for two of *the most thrilling inventions* in theoretical computer science in the 20th century: the Diffie-Hellman (DH) key exchange protocol (Whitfield Diffie and Martin Hellman, 1976) and the RSA scheme (Ron Rivest, Adi Shamir and Leonard Adleman, 1977). Both schemes draw their security from a certain "complexity gap" or "efficiency gap" in the current state of (classical) computational number theory, namely regardless of the efficiency of running them, no efficient algorithms are currently known for breaking DH and RSA.

2.1.1 Some number theory results

Definition 2.1. *A **multiplicative subgroup** is* $\mathbb{Z}_n^* := \{\, k \in \mathbb{Z}_n \mid \exists k^{-1} \,\}$.

Definition 2.2. ***Euler's phi-function or totient function*** *Let n be a natural number with prime factorization*

$$n = p_1^{k_1} \cdots p_m^{k_m}, \tag{2.1}$$

The Euler's phi-function is

$$\varphi(n) := p_1^{k_1-1}(p_1 - 1) \cdots p_m^{k_m-1}(p_m - 1). \tag{2.2}$$

$\varphi(n)$ gives the number of coprimes of n that does not exceed n. Hence $\varphi(n) = |\mathbb{Z}_n^*|$.

Theorem 2.3. ***(Gauss)*** *The multiplicative group $\mathbb{Z}_n^*$ is cyclic if and only if $n = 1, 2, 4$ or of the form p^k or $2p^k$ for an odd prime p. If $\mathbb{Z}_n^*$ is cyclic, it possesses $\varphi(\varphi(n))$ generators.*

Definition 2.4. *A prime p is called a* ***Sophie Germain prime*** *if $2p+1$ is also a prime. Then $2p+1$ is called a safe prime.*

Conjecture 2.5. *Let $\sigma(x)$ denote the number of Sophie Germain primes smaller or equal to x. Then it is conjectured that*

$$\sigma(x) \approx 1.320 \cdot \frac{x}{(\log x)^2} \tag{2.3}$$

Theorem 2.6. *Let $2p+1$ be a safe prime. Then a number $g \in \mathbb{Z}_{2p+1}$ is a generator of $\mathbb{Z}^*_{2p+1}$ if and only if*

$$g^2 \neq 1 \mod 2p+1 \text{ and } g^p \neq 1 \mod 2p+1 \tag{2.4}$$

This means generators of $\mathbb{Z}^*_{2p+1}$ is easy to compute for a safe prime $2p+1$, since

- $\varphi(\varphi(2p+1)) = p-1$, so the probability for a randomly chosen element to be a generator is around 1/2.
- Testing whether an element is a generator is efficient.

2.1.2 Some computational complexity results

Addition $x+y$ $O(n)$ for input numbers x, y of length n.

Multiplication $x \cdot y$ $O(n \log n)$ for input numbers x, y of length n.

Modular exponentiation $x^k \mod r$ **(double and add)** $O(n^2 m)$ for input number k of length m, r of length n.

Extended Euclid's algorithm (EEA) Find a, b such that $ax + by = \gcd(x, y)$.
Let $r_0 = x$, $r_1 = y$, $s_0 = 1$, $s_1 = 0$, $t_0 = 0$ and $t_1 = 1$. Repeat the procedure:

$$\begin{aligned} r_{i+1} &= r_{i-1} - q_i r_i,\ 0 \leq r_i < r_{i-1}; \quad \text{(which defines } q_i\text{)} \\ s_{i+1} &= s_{i-1} - q_i s_i; \\ t_{i+1} &= t_{i-1} - q_i t_i; \end{aligned} \tag{2.5}$$

and stop at $r_{i+1} = 0$. Then $r_i = \gcd(x, y) = s_i x + t_i y$. EEA is $O(n^3)$ for input numbers x, y of length n.
Note that if $\gcd(x, y) = 1$, $s_i = x^{-1}$ in $\mathbb{Z}^*_y$.

Deterministic primality testing Current best result is $O(n^{7.5})$ for input number x of length n.

Probabilistic primality testing of x Inputs: x of length n and $k \in \mathbb{N}$. Output: prime with probability $1 - 2^{-k}$ or no prime for sure.

Theorem 2.7. *Let x be an odd number and $x - 1 = 2^t m$ with odd m. If x is a prime number, then for every a coprime to x,*

$$a^m = 1 \mod x \qquad \text{or} \qquad \exists 0 \leq s < t : a^{2^s m} = -1 \mod x. \tag{2.6}$$

Conversely, assume x is not prime. Then the set A of coprimes $0 \leq a < x$ that fulfills the above conditions is small:

$$|A| \leq \varphi(x)/4. \tag{2.7}$$

Miller-Rabin test: Choose random $a \in \mathbb{Z}_x^*$ for $k/2$ times and do the test from the theorem. $O(kn^3)$.

Factorization / Discrete Logarithm Best known (classical) algorithm: $O(2^{\sqrt[3]{64/3}(\ln x)^{1/3}(\ln \ln x)^{2/3}})$ for input number x in the former and output x in the latter.

Some *provably not efficiently solvable* problems

- k-Halting Problem. Input: Turing program P, natural number k. Output: ”Yes” if P halts within k steps, ”No” otherwise.
- Halting Problem. Input: Turing program P. Output: ”Yes” if P halts, ”No” otherwise.

2.2 Diffie-Hellman key exchange

Whitfield Diffie and Martin E. Hellman invented asymmetric cryptography in 1976. The main steps of their protocol are:

- Choose a uniformly random safe prime $q = 2p + 1$ of length n with error probability 2^{-k}. The complexity is $O(kn^5)$ ($O(kn^3)$ for the Miller-Rabin test and $O(n^2)$ for the density of safe primes). Practically n is chosen to be 2000-3000 bits.

- Choose a uniformly random generator g of $\mathbb{Z}_q^*$. The complexity of a single test of generator is $O(n^3)$, and half of elements in $\mathbb{Z}_q^*$ are generators.
- The public information includes: a large safe prime q and a generator g of $\mathbb{Z}_q^*$. Alice (and respectively Bob) pick randomly $a \in \mathbb{Z}_q^*$ ($b \in \mathbb{Z}_q^*$ respectively) and send $g^a \mod q$ ($g^b \mod q$) to Bob (Alice).
- Now both Alice and Bob can compute the shared key $K = g^{ab} \mod q$ efficiently.

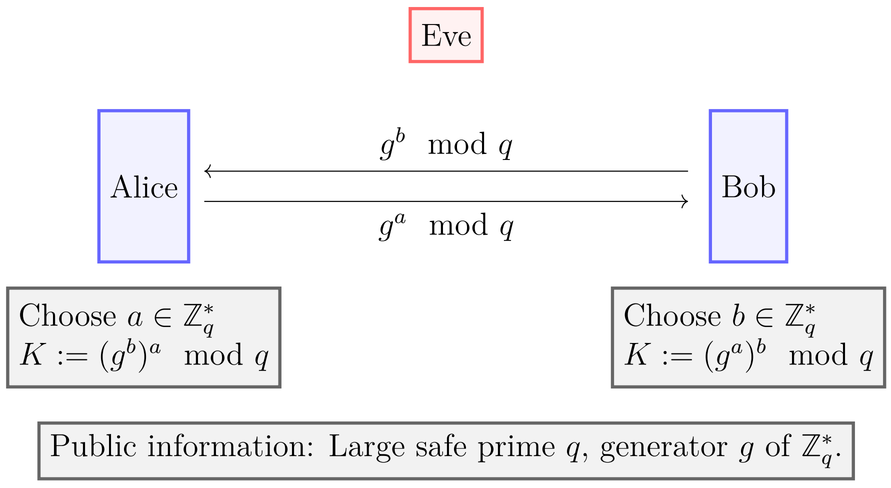

Eve
Alice
Bob
$g^b \mod q$
$g^a \mod q$
Choose $a \in \mathbb{Z}_q^*$
$K := (g^b)^a \mod q$
Choose $b \in \mathbb{Z}_q^*$
$K := (g^a)^b \mod q$
Public information: Large safe prime q, generator g of $\mathbb{Z}_q^*$.

Figure 2.1: Diffie-Hellman Key Exchange Protocol.

Security of Diffie-Hellman protocol

- *No* classical efficient algorithm for computing $g^{ab} \mod q$ from g^a and $g^b \mod q$ is known (Diffie-Hellman or DH-problem).
- *No* classical efficient algorithm for computing discrete logarithms is known.
- *Unlimited* computing power does help Eve!
- *Quantum computers* (once/if built) could solve DiscreteLog- and DH-problem efficiently.
- If you can efficiently solve DiscreteLog, then you can break DH-problem. They are not however generally proven to be equivalent.

Thus there might be (but unknown) smarter ways to attack DH key exchange.

- Many implementations use fixed prime q or choose their primes q from limited lists. This can however cause unwanted side effects! It is better to choose and use your own, personal, fresh prime.

Whys

- Why do we calculate in $\mathbb{Z}_q^*$? *Otherwise g^a would be extremely large and the logarithm can be estimated from its length.*
- Why do we use a generator g instead of a general element x? *If x happens to have a small span, then eve can compute the table of $x^i \mod q$ in prior.*
- Why do we use safe primes? *For safe primes, there are many generators (half) and they are easy to be verified. And the baby-step-giant-step (BSGS) algorithm can compute $\log_g g^a \mod N$ efficiently for a generator g of N if N solely has small prime factors; it will not work for Diffie-Hellman key exchange.*

Man in the middle attack Unfortunately, a smart Eve can still steal the key. Instead of just passively eavesdropping the communication "from outside", Eve gets active, and interferes with / alters communication!

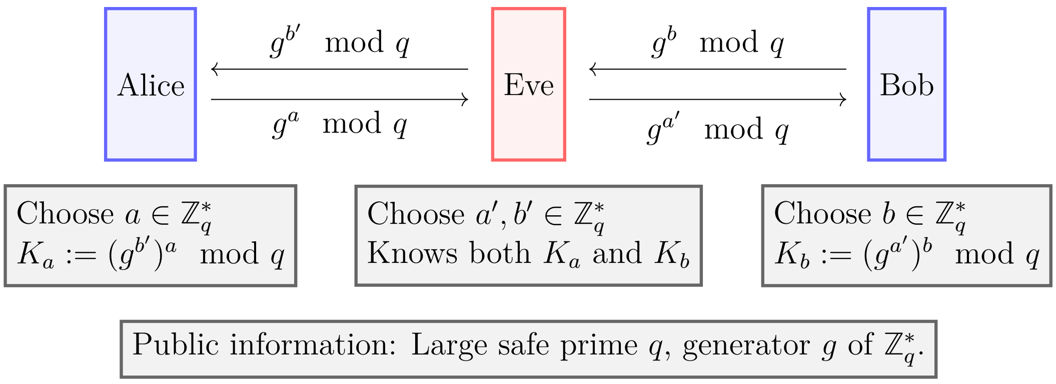

Alice
$g^{b'} \mod q$
$g^a \mod q$
Eve
$g^b \mod q$
$g^{a'} \mod q$
Bob
Choose $a \in \mathbb{Z}_q^*$
$K_a := (g^{b'})^a \mod q$
Choose $a', b' \in \mathbb{Z}_q^*$
Knows both K_a and K_b
Choose $b \in \mathbb{Z}_q^*$
$K_b := (g^{a'})^b \mod q$
Public information: Large safe prime q, generator g of $\mathbb{Z}_q^*$.

Figure 2.2: Man in the middle attack.

See Fig. 7.1, once Eve has established key K_a with Alice and K_b with Bob, she can forever act as an unnoticed "man in the middle",reading all messages! The countermeasure is to *authenticate* the communication chancel, which is *indeed necessary* for any secure key exchange protocol: Diffie-Hellman, quantum key exchange, etc. Possibilities for authentication includes: trusted carrier, trusted postal delivery, small initially shared key (key exchange works as a ”key amplification / agreement” scheme), etc.

2.3 RSA scheme

The RSA scheme is another asymmetric encryption protocol. The setup includes initialization (generating public and private keys), encryption (using only public key) and decryption (using only private key).

Initialization of RSA scheme

- Choose independently and uniformly at random two large primes p and q of roughly same size n at error probability 2^{-k}. Again we use the Miller-Rabin test and the complexity is $O(kn^4)$
- Compute $N = pq$ and $\varphi(N) = (p-1)(q-1)$.
- Choose an encryption exponent e that is coprime to $\phi(N)$; a popular choice is $e = 2^{16} + 1 = 65537$.
- Compute decryption exponent $d = e^{-1}$ in $\mathbb{Z}_{\varphi(N)}$. Complexity is $O(n^3)$ with EEA algorithm. Without knowing p, q and $\varphi(N)$, computing d is hard.
- Set *the public key* to be $PK = (e, N)$.
- Set *the private key* to be $SK = (d, N)$.

Encryption and Decryption with RSA Represent message M as number(s) in $\mathbb{Z}_N$.
Encryption:

$$\text{Ciphertext} = M^e \mod N \tag{2.8}$$

Decryption:

$$\begin{aligned} \text{Plaintext} &= (\text{Ciphertext})^d \mod N \\ &= (M^e)^d \mod N \\ &= M \mod N. \end{aligned} \tag{2.9}$$

See Proof. 2.3

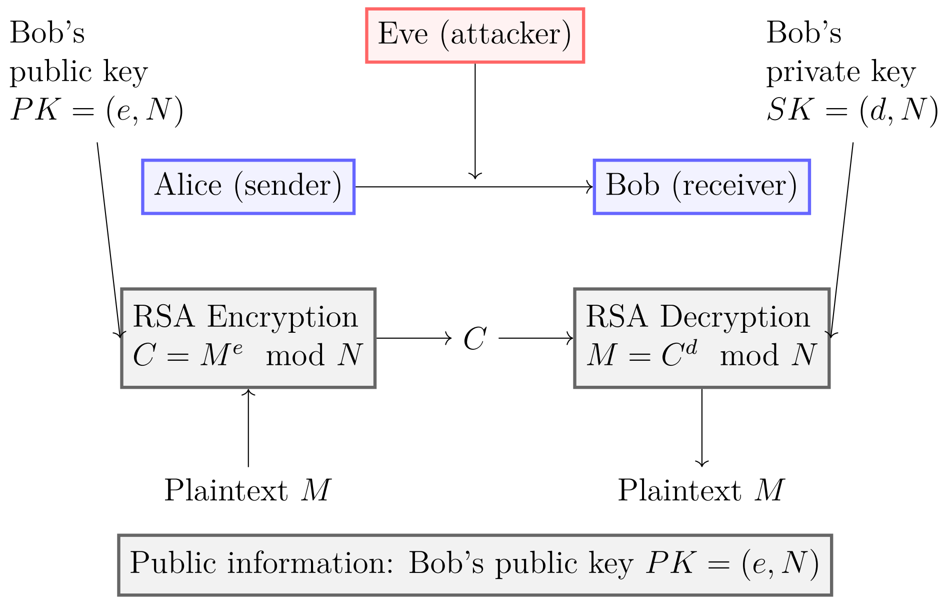

Bob's public key $PK = (e, N)$
Eve (attacker)
Bob's private key $SK = (d, N)$
Alice (sender)
Bob (receiver)
RSA Encryption $C = M^e \mod N$
C
RSA Decryption $M = C^d \mod N$
Plaintext M
Plaintext M
Public information: Bob's public key $PK = (e, N)$

Figure 2.3: RSA scheme.

Theorem 2.8. ***RSA-Decryption Inverts RSA-Encryption.*** *Let (e, N) be a public key of RSA and (d, N) be the associated private key. Then it holds for all $M \in \mathbb{Z}_N$ that*

$$(M^e)^d = M \mod N \tag{2.10}$$

Proof. As $ed = 1 \mod \varphi(N)$, $ed = 1 + k(p-1)(q-1)$, $k \in \mathbb{Z}$. Therefore

$$(M^e)^d \quad = M(M^{(p-1)(q-1)})^k = M \mod p \tag{2.11}$$

and

$$(M^e)^d \quad = M(M^{(p-1)(q-1)})^k = M \mod q \tag{2.12}$$

according to Fermat's little theorem. Hence

$$(M^e)^d = M \mod pq = M \mod N. \tag{2.13}$$

□

Private-key vs. public-key cryptography in large networks

Asymmetric crypto requires *much fewer* keys (N for N participants) in large networks compared with the symmetric case ($N(N-1)/2$).

Practical Security of RSA The security of RSA is related to the *long-standing unsolved factoring problem.* Some facts are

1. RSA can be broken with efficient factorization algorithm.
2. Directly computing the decryption component d is equivalent to factorizing N. $\Leftarrow$ is trivial, while $\Rightarrow$ relies on the Chinese remainder theorem.

   Given (e, d), let $k = de - 1$, which is a multiple of $\phi(N)$. Since $\phi(N)$ is even, $k = 2^t r$ with r odd for some $t \geq 1$. $g^k = 1 \mod N$ for any $g \in \mathbb{Z}_N^*$, and hence $g^{k/2}$ is a square root of unity. By Chinese remainder theorem, 1 has four square roots modulo $N = pq$: ± 1 and $\pm x$, where $x = 1 \mod p$ and $x = -1 \mod q$. Given x, one can reveal $p = \gcd(x - 1, N)$. It can be shown that for randomly chosen g, the probability of one of $k/2, \cdots k/2^t$ being $\pm x$ is at least $1/2$.
3. As a consequence, different parties can not share the same modulus N. Otherwise any of them can factorize N by (e, d)-pair. So the global RSA setup that a trusted party chooses global p, q, N and distribute (e_i, d_i) to different participants does not work.
4. There could be other, ”smarter” ways of breaking RSA than factoring N, but we currently can neither formally prove or disprove it.
5. In any case, RSA only has *computational security.* In 1977, inventors published RSA-modulus N with 129 digits, which is considered to take 40 quadrillion (10^{15}) years to break with the fasted computer at that point. In 1994, though, RSA-129 was factored by an improved factoring algorithm and massively parallel computations distributed all over the internet. Today the recommended key lengths by NIST is 2048-bit until 2030 and 3072-bits beyond.

RSA today: Optimal Asymmetric Encryption Padding (OAEP, Bellare and Rogaway, 1993) In RSA, the plaintext should not be deterministic; otherwise eve can simply encrypt all the possible plaintexts and compare them with the ciphertexts. OAEP is aimed to make the plaintext probabilistic.
Assume encryption / decryption function Enc_{PK} and Dec_{SK} for k-bit strings with $k = n + k_1 + k_0$, where n is the length of message, k_1 is the length of checking bits and k_0 is the length of security parameters that are infeasible for the adversary. Assume two hash functions

$$G : \{0,1\}^{k_0} \to \{0,1\}^{k-k_0}; \qquad H : \{0,1\}^{k-k_0} \to \{k_0\}. \tag{2.14}$$

Finally, for plaintext $M \in \{0,1\}^n$:

1. Choose uniformly random $r \in \{0,1\}^{k_0}$;
2. Compute $s = (M||0^{k_1}) \oplus G(r)$, $t = r \oplus H(s)$;
3. Ciphertext $C = \mathrm{Enc}_{PK}(s||t)$.

In order to decrypt,

1. Recover s and t by $s||t = \mathrm{Dec}_{PK}(C)$;
2. Recover $r = t \oplus H(s)$ and $M' = s \oplus G(r)$
3. If $[M']_{k_1} = 0^{k_1}$, output $[M']_n$; otherwise reject the input ciphertext.

2.4 Asymmetric identification

As discussed in MAC, even if Eve cannot understand or decrypt the ciphertext, she may still modify it. There is also an asymmetric version of identification to ensure message integrity.

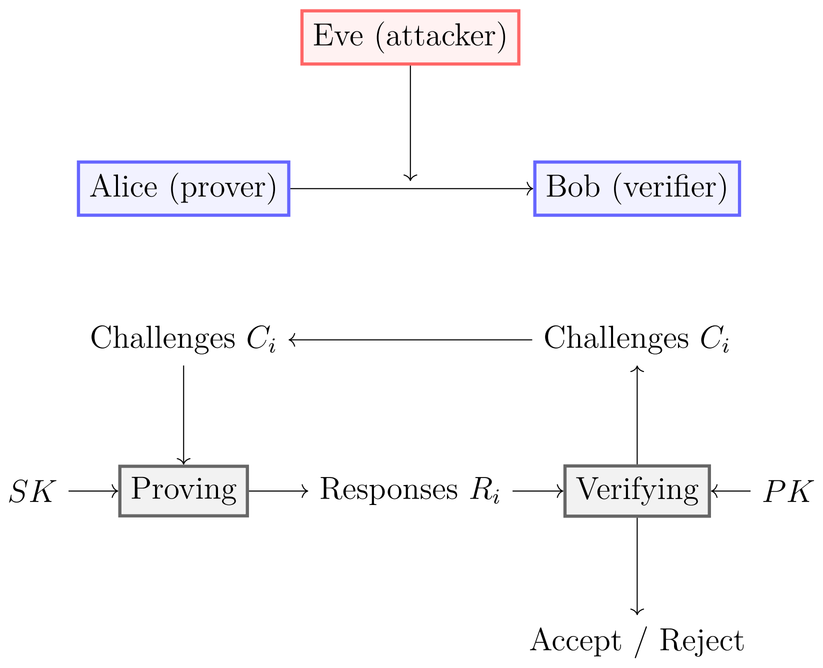


Figure 2.4: Asymmetric identification.

- Eve is assumed to know: general method of proving and verifying and of choosing keys, public key of verifier and previous challenge / response pairs. Eve can also actively interact with prover as ”verifier in disguise” several times.
- Eve’s attack target: learn anything about the private key SK of Alice and eventually make the verifying method return ”Accept” when talking to Eve.

Digital signatures One way of asymmetric identification is to use digital signatures. It mainly follows the same idea as public-key encryption, but just encrypts with private key and decrypts with public key instead. In practice usually the hashed value Hash(M) of the message M is signed by RSA. This choice guarantees:

- efficiency, since only a single block instead of the whole message is encrypted;
- security, since it prevents the blinding attack: Eve can pretend a message M as $M^* = r^e M \mod N$ and compute $r^{-1} \mod N$ for some random $r \in \mathbb{Z}_N^*$. Then M^* looks random, but, if it is signed, Eve can recover the encrypted message of M by $M^d = (M^*)^d r^{-1} \mod N$! Hash function however, destroy the isomorphic multiplicative property of RSA;
- also it prevents changing the sequence of the blocks.

Bob sends each time a random nonce R_n and Alice encrypts it with the digital signature. Note that do not use the same (SK, PK)-pair for both identification protocols and legally binding signatures.

Note

No digital signatures with information-theoretic security are possible.

Chapter 3

Advanced topics

3.1 Two players protocol

Definition 3.1. *A* ***cryptographic protocol*** *is an abstract or concrete protocol that performs a security-related function and applies cryptographic methods, often as sequences of cryptographic primitives. A protocol describes how the algorithms should be used.*

Motivation Alice and Bob want to determine who gets \$100 without splitting them. Both of them have a fair coin, but neither trusts the other. What can they do?

1. (Commit.) Alice tosses her coin and gets 1 or 0. She writes down the result and sends it to Bob in a locked box.
2. (Reveal and check.) Bob tosses his coin and openly tells Alice his result. Alice sends her key to Bob and Bob opens the box. Now both can XOR the results and get the same number.

Definition 3.2. *(****Perfect bit commitment****.) A function comm* $: \{ 0, 1 \} \times X \to Y$*, where* X *is a set of keys, is called a perfect bit commitment if* X *and* Y *are publicly known sets so that for* $(b, x) \in \{ 0, 1 \} \times X$*, a party* P *can send* $y = comm(b, x)$ *to a party* $V \neq P$ *with the following conditions:*

- *Revealing:* V *can determine* b *from* (x, y)*.*
- *Efficiency: all computations can be done in polynomial time with regard to security parameter* n*.*
- *Computational concealing: it is hard for* V *to determine* b *from* y *alone.*

- *Perfect binding: there is no message $x' \in X$ such that comm$(b \oplus 1, x') = y$.*

Relaxation

Perfect binding can be relaxed to computational binding, i.e., it is hard to find such x'; and such a function is called *computational bit commitment.*

Examples of bit commitment functions

- Hashing: $y = H(b||x)$. Send firstly only y and afterwards (b, x). Bob can only check that b satisfies the Hash function, but not be able to compute b from (x, y). Also a hash function is only computational binding.
- RSA: choose $N = pq$ and e like in RSA; choose $1 \leq x_0 \leq (N-1)/2$ and $x = 2x_0 + b \mod N$. $y = x^e \mod N$. Send firstly (N, e, y) and afterwards (p, q, b, x). In the checking step, verify that $N = pq$, $e \in \mathbb{Z}^*_{\phi(N)}$, $y = x^e \mod N$ and that $b = x \mod 2$. It is a perfect bit commitment for now.
- Quadratic residues: pick $q \in \mathbb{Z}^*_N$ which is known to be a non-residue and randomly choose x. Send firstly (N, q, y) where $y = x^2 \mod N$ if $b = 0$ and $y = qx^2 \mod N$ if $b = 1$. Bob cannot efficiently distinguish them if N is a composite whose factorization is unknown. It is a perfect bit commitment for now as well.
- Discrete logarithm: let p be a prime, $g \in \mathbb{Z}^*_p$ be a generator and pick s such that $\log_g s$ is unknown. Choose random x; send firstly (p, g, s, y) where $y = g^x \mod p$ if $b = 0$ and $y = sg^x \mod p$ if $b = 1$. It is not revealing, but Alice can cheat only when she knows $\log_g s$, which cannot be done efficiently.

3.2 Oblivious transfer (OT)

Rabin-OT Alice is a researcher. For a study, she wants half of the people to think about something and the other half to think about whatever they want. But Alice shouldn't know who is who, or she will be biased. Also, none of the participants should know what others think about. What can Alice do?

- Input: Alice has some information i while Bob may have some random bit r.

- Output: Bob obtains i with probability 1/2, possibly depending on r.
- Security requirements: Alice *cannot* learn whether Bob received i.

1-out-of-2-OT Alice is a researcher. For a study, Bob should think of one of two words depending on whether or not he earned $\geq$ 450€ last month. But Alice shouldn't know Bob's income. What can Alice do?

- Input: Alice holds two secrets s_0 and s_1 while Bob holds a choice bit b.
- Output: Bob obtains the secret s_b.
- Security requirements: Bob *cannot* learn both secrets s_0 and s_1 and Alice cannot learn Bob's choice bit b.
- Bob can learn the wrong bit if he isn't honest. But in that case he can't learn the right one.

These two flavors of OT are in fact *computationally equivalent.* We propose a 1-out-of-2-OT with a discrete logarithm in Fig. 3.1.

$$\boxed{\text{Public information: prime} p,\ g \in \mathbb{Z}_p^* \text{ whose logarithm is unknown.}}$$

$$\begin{array}{ccc}
\text{Alice (Sender)} & & \text{Bob (Receiver)} \\
c \in \mathbb{Z}_p^* & \xrightarrow{c} & \\
 & & x \in \mathbb{Z}_p \\
 & \xleftarrow{h_0} & h_b = g^x,\ h_{1-b} = c/h_b \\
h_1 = c/h_0 & & \\
k \in \mathbb{Z}_p,\ c_1 = g^k & & \\
e_0 = s_0 \oplus H(h_0^k) & & \\
e_1 = s_1 \oplus H(h_1^k) & \xrightarrow{c_1, e_0, e_1} & \\
 & & s_b = e_b \oplus H(c_1^x)
\end{array}$$

Figure 3.1: 1-out-of-2-OT with a discrete logarithm.

OT is a *universal cryptographic primitive*, i.e., it allows the implementation of a very large number of other cryptographic protocols, including bit commitment, zero-knowledge proofs, secure multiparty computation, etc. It has thus often been used as a first touchstone for the potential of new cryptographic models:

- Quantum crypto (no);

- Noise-based crypto (yes);
- Strong physical unclonable Functions (PUFs) (yes).

Examples

Ito-Nishikezi-Saito secret sharing There are two qualifying sets to reconstruct the secret s: $(PlandG) \vee (G \wedge V \wedge S)$. The construction is to generate five bitstrings s_i such that

$$s = s_P \oplus s_G^1 = s_G^2 \oplus s_V \oplus s_S. \tag{3.1}$$

It can be shown that any of the qualifying sets and only the qualifying sets recover s.

Polynomial interpolation A polynomial p of degree n is uniquely determined by $n+1$ distinct values $p(x_1), \cdots, p(x_{n+1})$. The $p(x_i)$ are called supporting points, which can be distributed to share secrets.

The millionaires' problem (Yao, 1982) Two millionaires want to determine who has more money, but without disclosing their wealth.
It there is no trusted third party, it can still be done by *secure* multi-party computation, i.e., the information which is leaked is precisely that which would have been leaked if the computation had been conducted by encrypting messages to a trusted third party. There are two types of adversaries:

- honest-but-curious: the parties are interested in breaking privacy, but follow the protocol with honest inputs. For instance, several of them may combine their data.
- malicious: they don‘t necessarily follow the protocol and may want to pass around incorrect data.

There are perfectly secure and computationally secure schemes. They can be secure even if half of the participants are honest-but-curious adversaries; computationally secure schemes also work for half of malicious participants, while perfectly secure does the job only for no more than 1/3.

3.3 Zero knowledge proof (ZKP)

Definition 3.3. *An* ***interactive proof system*** *consists of two interacting algorithms Peggy (prover) and Victor (verifier). They get the same input but use separate memory and random strings. Peggy is omnipotent (having unlimited power) and Victor uses random polynomial time. When one of the two parties has finished its computations, it sends a message to its counterpart and switches to waiting mode. Then it is the other's turn.*
Victor decides at the end whether he accepts or rejects, and outputs the result "accept" or "reject". The system is an interactive proof system for a problem $X \subseteq \{0,1\}^*$ *if it is:*

- *complete: Victor accepts every* $x \in X$ *with probability at least* $2/3$ *(this rate may be replaced by and constant larger than* $1/2$*);*
- *sound: for any algorithm that Peggy might use and for any* $x \in \{0,1\}^* \setminus X$*, Victor accepts* x *with probability at most* $1/3$*.*

Definition 3.4. *Let* $G_1 = (V_1, E_1)$ *and* $G_2 = (V_2, E_2)$ *denote two graphs. A* ***graph isomorphism*** *is a bijection* $\pi : V_1 \to V_2$ *such that*

$$\forall v, w \in V_1, (v,w) \in E_1 \Leftrightarrow (\pi(v), \pi(w)) \in E_2. \tag{3.2}$$

We write $G_1 \cong G_2$*.*

Graph isomorphism problem Given two graphs G_1, G_2 on the same vertex-set $1, \cdots, n$, decide: are they isomorphic?
This problem now lies in the complexity class $NP\backslash P$, which means it is presumably hard to solve a random problem instance but easy to verify a given solution. Say, Peggy knows an isomorphism π between two graphs G_1 and G_2. She could easily convince Victor that $G_1 \cong G_2$ by telling him π. But can she convince him *without* revealing anything about π?
The answer is *yes*. Peggy and Victor can perform the IP procedure described in Fig. 3.2 for n rounds and Victor returns accept only if all rounds are satisfactory. Property check of the IP with graph isomorphism:

- Completeness: If Peggy knows the isomorphism, she can always compute correctly and Victor will accept each round.

$\boxed{\text{Public info: Two graphs } G_1 \text{ and } G_2 \text{ where } V_1 = V_2 = \{1, \cdots, n\}}$

$\boxed{\text{Peggy's secrete: a graph isomorphism } \pi : G_1 \to G_2}$

Peggy (Prover)		Victor (Verifier)
Randomly choose a permutation σ of $\{1, \cdots, n\}$. $H := \sigma(G_2)$	$\xrightarrow{H}$	
	$\xleftarrow{i}$	Pick $i \in \{0, 1\}$
Compute $\rho = \sigma$ if $i = 1$ and $\sigma \circ \pi$ if $i = 0$	$\xrightarrow{\rho}$	
		Check if $\rho(G_i) = H$

$\boxed{\text{Execute } n \text{ rounds}}$

Figure 3.2: Interactive proof (IP) with graph isomorphism

- Soundness: If $G_1 \ncong G_2$, H is isomorphic to at most one of them and the final acceptance rate would be only $1/2^n$.
- *Zero-knowledge*: Victor learns nothing about π, since he only knows a random permutation σ or $\sigma \circ \pi$.

When we talk about ZKP, in particular Victor should not be able to convince anyone else that he knows the secret of Peggy. Assume that everything Victor knows is in the message, or a *transcript* of everything Victor knows:

Definition 3.5. *A* ***transcript*** *of an execution of an interactive protocol is the list of all inputs and messages exchanged.*

Victor gained no non-trivial knowledge if a ”simulator Sam” (Goldwasser & Micali, 1989) can produce a simulation-transcript that is indistinguishable from a real one. In the procedure above, the transcript consists of (G_0, G_1) and (H_n, i_n, ρ_n) of each enquiry, which can be made without any knowledge of π. Peggy can convince Victor only through the ”interactive” process.

Definition 3.6. *An interactive proof system Peggy / Victor for a problem X has* ***perfect zero knowledge*** *if there exists a random polynomial-time simulator Sam which on any input x (= a problem instance) produces a transcript $T_{SAM}(x)$ such that*

$$Prob(T_{P,V}(x) = T) = Prob(T_{SAM}(x) = T) \tag{3.3}$$

for any conceivable transcript T.

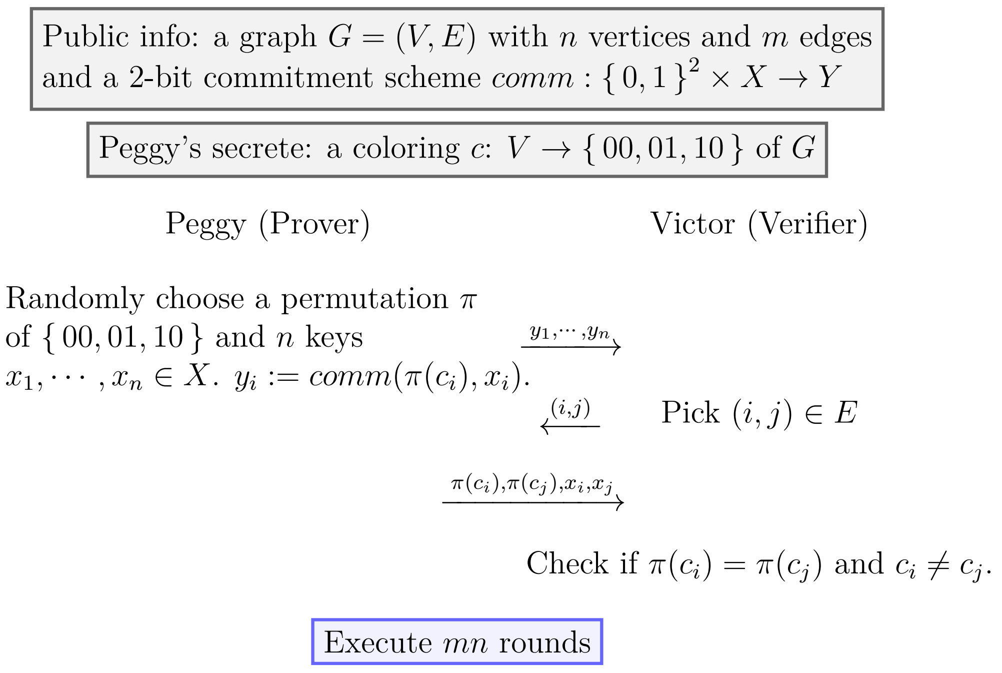
Theorem 3.7. *The graph isomorphism protocol has perfect zero knowledge.*

Definition 3.8. *An interactive proof system Peggy / Victor for a problem X has* ***computational zero knowledge*** *if we require only the probability distribution to be polynomially indistinguishable.*

Computational ZKP from graph 3-coloring

Definition 3.9. *A k-**coloring** of a graph $G = (V, E)$ is a map $c : V \to \{1, \cdots, k\}, i \mapsto c_i$ such that $\forall (i, j) \in E$, $c_i \neq c_j$. If such a coloring exists, we say G is k-**colorable**.*

A computational ZKP scheme from graph 3-coloring is shown in Fig. 3.3.



Public info: a graph $G = (V, E)$ with n vertices and m edges and a 2-bit commitment scheme $comm : \{0, 1\}^2 \times X \to Y$

Peggy's secrete: a coloring c: $V \to \{00, 01, 10\}$ of G

Peggy (Prover) Victor (Verifier)

Randomly choose a permutation π of $\{00, 01, 10\}$ and n keys $x_1, \cdots, x_n \in X$. $y_i := comm(\pi(c_i), x_i)$. $\xrightarrow{y_1, \cdots, y_n}$

$\xleftarrow{(i,j)}$ Pick $(i, j) \in E$

$\xrightarrow{\pi(c_i), \pi(c_j), x_i, x_j}$

Check if $\pi(c_i) = \pi(c_j)$ and $c_i \neq c_j$.

Execute mn rounds

Figure 3.3: Computational ZKP from graph 3-coloring

Again we do the property check:

- Completeness: if Peggy knows a coloring, she can commit a suitable coloring each time and Victor will accept each round.
- Soundness: if the graph is not 3-colorable, at least one of the m edges has $c_i = c_j$. Victor chooses this edge with probability $1/m$ in each round and is fooled with total probability $(1 - 1/m)^{nm}$.

- Zero-knowledge: Victor learns nothing about the coloring, since he only sees two randomly permutated colors in each round and can not collect the information.
- Perfect zero knowledge: unknown. The system is computationally zero-knowledge, however.

As k-color problem is NP-complete for any $k \geq 3$, this theorem follows:

Theorem 3.10. *All problems in NP have a computational zero knowledge proof system.*

3.4 Fully homomorphic encryption

Motivation Real-time navigation might be too much for a constrained device. Doing it on cloud might however leak the destination to Eve. Alice thus wants the data on cloud always encrypted during the computation. Goal: results of encrypted processing = results of clear-text processing!

Definition 3.11. ***(Fully homomorphic encryption scheme, FHE.)*** *Let C be a class of circuits $c : \{0,1\}^n \to \{0,1\}$. An encryption scheme (KeyGen, Enc, Dec, Eval) is called $C-$**homomorphic** if $\forall f \in C$ and ciphertexts $c_1, \cdots, c_n$, it holds that $Eval(f, c_1, \cdots, c_n) = c^*$ such that*

$$\mathrm{Dec}(c^*) = f(\mathrm{Dec}(c_1), \cdots \mathrm{Dec}(c_n)). \tag{3.4}$$

*An encryption scheme is **fully homomorphic** if it is P-homomoprhic, where P is the class of all polynomial sized circuits.*

Most currently known FHE schemes are very power- and / or memory-consuming. For instance, Microsoft's SEAL (Simple Encrypted Arithmetic Library):

- 1 MB data $\simeq$ 10 GB encrypted data;
- Addition takes $\sim$ 1 msms;
- Multiplication takes $\sim$ 5 s!

Even with new technologies, FHE will probably stay useful only for certain areas.

3.5 Elliptic curve cryptography

Definition 3.12. ***(Elliptic curve.)*** *Let F be a field of characteristics other than 2 and 3 and $a, b \in F$ with $4a^3 + 27b^2 \neq 0$. Then*

$$E = \{\, (x, y) \in F^2 : y^2 = x^3 + ax + b \,\} \cup \{\, \mathcal{O} \,\} \subseteq F^2 \cup \mathcal{O} \tag{3.5}$$

is an elliptic curve over F. $\mathcal{O}$ denotes the infinity point on E.

Elliptic curves are of interest for cryptography, as they naturally carry a (commutative) cyclic group structure:

- The identity is the infinite point $\mathcal{O}$.
- The inverse element of $P = (x, y)$ is its mirror image along the x-axis $-P = (x, -y)$. Note that $-\mathcal{O} = \mathcal{O}$.
- The group operation $+$ is such that $P + Q = R := -S$, where S is the intersection of of a line through P and Q with E. If $P = Q$, then we take the tangent of P and its intersection with E; if $Q = \mathcal{O}$, then we take the vertical line through P. The line through the points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ on $E : y^2 = x^3 + ax + b$ is given by

$$y = m(x_2 - x_1) + y_1, \text{ where } m = \begin{cases} (y_2 - y_1)/(x_2 - x_1), & P \neq Q; \\ (3x_1^2 + a)/2y_1, & P = Q; \end{cases} \tag{3.6}$$

- Multiplying an integer $k \in \mathbb{Z}$ with P means applying the group operation exactly k times on P, while $0P = \mathcal{O}$.

For cryptographic purposes, we usually prefer elliptic curves with $F = \mathbb{F}_q$ (q is prime).

A playground example

$$E : y^2 = x^3 - x + 1 \text{ in } \mathbb{Z}_7. \tag{3.7}$$

It contains 12 points: $(0,1)$, $(0,6)$, $(1,1)$, $(1,6)$, $(2,0)$, $(3,2)$, $(3,5)$, $(5,3)$, $(5,4)$, $(6,1)$, $(6,6)$ and $\mathcal{O}$. 12 is not a prime, otherwise all points would be generators; in this case $G = (1,6)$ is a generator of E.

Hard problems in elliptic curves and pairing

1. Problem: Elliptic Curve DiscreteLogarithm *(ECDiscLog).*
   - Input: elliptic curve $E(\mathbb{F}_q)$; generator P of E; value xP.
   - Output: x.
2. Problem: Elliptic Curve Diffie-Hellmann *(ECDH)*
   - Input: elliptic curve $E(\mathbb{F}_q)$; generator P of E; values aP, bP.
   - Output: $(ab)P$.
3. Problem: Decisional Diffie-Hellmann problem
   - Input: elliptic curve $E(\mathbb{F}_q)$; generator P of E; $a, b, z \in \mathbb{Z}_p^*$; values: aP, bP, zP.
   - Output: True if $ab = z$ and False otherwise.

We will introduce the pairing functions that allow constructions where the EDDiscLog and ECDH problems are *computationally hard* but the decision Diffie-Hellmann problem is *easy.* If we use the elliptic curves instead, the key-exchange, digital signatures and public key encryption stay the same; the weaknesses, e.g., against quantum computers however also stay the same.

Symmetric Key	Dlog / RSA	ECDLog
80	1,024	160
128	3,072	256
192	8,192	384
256	15,360	512

Table 3.1: Common measures on key sizes.

Definition 3.13. *A **pairing** is a bilinear map on a group M taking values in another group R:*

$$\langle \cdot, \cdot \rangle : M \times M \to R \tag{3.8}$$

supposing that M is an addition group and R is a multiplicative group. The bilinearity means

$$\begin{aligned} \langle x + y, z \rangle &= \langle x, z \rangle \cdot \langle y, z \rangle \, ; \\ \langle x, y + z \rangle &= \langle x, y \rangle \cdot \langle x, z \rangle \, . \end{aligned} \tag{3.9}$$

For cryptographic purposes, we denote a pairing as follows:

Definition 3.14. *Let $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ be cyclic groups (we typically use additive elliptic curve groups $E(\mathbb{F}_q)$ for $\mathbb{G}_1$ and $E(\mathbb{F}_{q^l})$ for $\mathbb{G}_2$, and a multiplicative group $\mathbb{F}^*_{q^k}$ for $\mathbb{G}_T$). A pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ has the following properties:*

- *(Bilinearity)* $\forall (P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2, \forall a, b \in \mathbb{Z}$: $e(aP, bQ) = e(P, Q)^{ab}$;
- *(Non-degeneracy)* $\forall (P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$: $e(P, Q) = \mathbb{1} \Leftrightarrow (P = \mathcal{O}) \vee (Q = \mathcal{O})$;
- *(Computability)* $\forall (P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$, *there is an efficient program to compute* $e(P, Q)$.

If $l = 1$, the pairing is symmetric; otherwise it is asymmetric with embedding degree l. We require arithmetic in $\mathbb{F}^*_{q^k}$, so k is wanted to be small for computability; for security, however, k must be large enough for DiscLog to be computationally hard.

Identity based signatures (The Hess Scheme)

- Public information:
  - $\mathbb{G}, P$: $E(\mathbb{F}_q)$ generated by P;
  - $\mathbb{G}_T, g$: $\mathbb{F}^*_q$ generated by g;
  - $e(P, P)$: symmetric pairing;
  - $H()$: hash function that maps to $\mathbb{G}$;
  - $h()$: hash function that maps to $\{0, 1\}^*$.
- Setup: random master key pair:
  - $msk \leftarrow \mathbb{Z}_p$ (secret in Bank);
  - $MPK \leftarrow mskP$ (public).
- Step 1: (Bank) key generation for id (of Alice, signer). Calculate $U = msk \cdot H(id)$ and send it to Alice.
- Step 2: (Alice, signer) sign a message m:
  - choose random: $x \leftarrow \mathbb{Z}_p$ and $Q \leftarrow \mathbb{G}$;
  - compute message hash: $h_m = h(m, e(P, Q)^x)$;

  - compute signature: $S = h_m U + xQ$.

  Send (m, h_m, S) to Bob, verifier.

- Step 3: (Bob, verifier) verify a signature:
  - compute $r = e(S, P) \cdot e(H(id), -MPK)^h$;
  - verify if $h_m = h(m, r)$.

The verification returns 1, if and only if $r = e(P, Q)^x$ where x and Q are unknown for the verifier. It is true as

$$\begin{aligned} r &= e(S, P) \cdot e(H(id), -MPK)^h \\ &= e(h_m U + xQ, P) \cdot e(H(id), -MPK)^h \\ &= e(h_m \cdot mskH(id) + xQ, P) \cdot e(H(id), -mskP)^h \\ &= e(h_m \cdot mskH(id), P) \cdot e(xQ, P) \cdot e(H(id), -mskP)^h \\ &= e(H(id), P)^{(h_m - h) \cdot msk} \cdot e(Q, P)^x. \end{aligned} \tag{3.10}$$

Identity based encryption (Boneh-Franklin Scheme)

- Public information:
  - $\mathbb{G}, P$: $E(\mathbb{F}_q)$ generated by P;
  - $\mathbb{G}_T, g$: $\mathbb{F}_q^*$ generated by g;
  - $e(P, P)$: symmetric pairing;
  - $H()$: hash function that maps to $\mathbb{G}$;
  - $h()$: hash function that maps to $\{0, 1\}^*$.
- Setup: random master key pair:
  - $msk \leftarrow \mathbb{Z}_p$ (secret in Bank);
  - $MPK \leftarrow mskP$ (public).
- Step 1: (Bank) key generation for id (of Bob, receiver). Calculate $U = msk \cdot H(id)$ and send it to Bob.
- Step 2: (Alice, sender) encrypt a message m:
  - choose random: $x \leftarrow \mathbb{Z}_p$ and $Q \leftarrow \mathbb{G}$;
  - compute: $C_1 = xP$;
  - compute: $C_2 = h(e(H(id), MPK)^x) \oplus m$.

Send (C_1, C_2) to Bob, receiver.

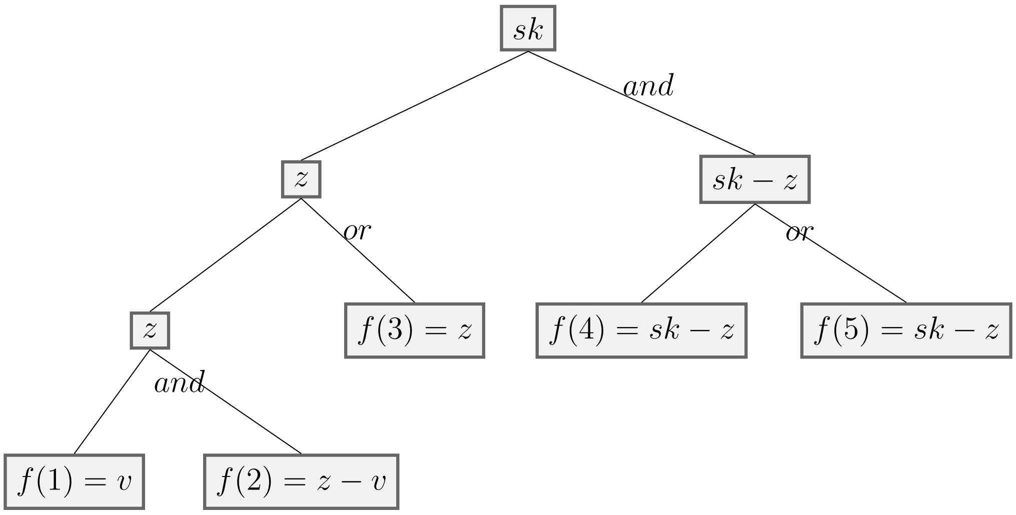
- Step 3: (Bob, receiver) decrypt: compute $m = h(e(U, C_1)) \oplus C_2$.

The decryption is successful if and only if $e(U, C_1) = e(H(id), MPK^x)$, which is correct as

$$\begin{aligned} e(H(id), MPK)^x &= e(H(id), mskP)^x \\ &= e(mskH(id), xP) = e(U, C_1). \end{aligned} \tag{3.11}$$

Attribute based encryption (GPSW06) The idea is that the president Alice stores data in the cloud storage and encrypts it with her encryption key. Others can decrypt with keys holding the access attributes.

- Attributes: set $U_n = \{1, \cdots, n\} \in \mathbb{Z}_p^*$.
- Public information:
  - $\mathbb{G}, P$: $E(\mathbb{F}_q)$ generated by P;
  - $\mathbb{G}_T, g$: $\mathbb{F}_q^*$ generated by g;
  - $e(P, P)$: symmetric pairing.
- Setup: random master key
  - $t_i \leftarrow \mathbb{Z}_p^* \forall i \in U$ and $sk \leftarrow \mathbb{Z}_p^*$ (secret of Alice);
  - $T_i = t_i P \forall i \in U$ and $PK = e(P, P)^{sk}$ (public).
- Step 1: (Alice) generate key share function f, e.g. $,v, z \leftarrow \mathbb{Z}_p^*$ and



- Step 2: (Alice) encrypt a message $m \in \mathbb{G}_T$:
  - choose a set of decryption attributes $\gamma \leftarrow U$;
  - choose random: $x \leftarrow \mathbb{Z}_p^*$;
  - compute: $C = (\gamma, C' = M \cdot PK^s, xP, \{\, C_i'' = T_i^S \,\}_{i \in \gamma})$.
- Step 3: (Alice) generate the key for Bob's attribute set L:
  - Bob's key share: $f(L)$;
  - choose random $\{\, r_i \leftarrow \mathbb{Z}_p^* \,\}_{i \in L}$;
  - compute the decryption key:
    $D_{\text{Bob}} = \{\, D_i' = f(i)P + r_i T_i, R_i = r_i P \,\}_{i \in L}$.

  Send (C, D_{Bob}) to Bob.
- Step 4: (Bob) decrypt:
  - compute $e(P,P)^{sk \cdot x} = \prod_{i \in L} e(X, D_i') / \prod_{i \in L} e(R_i, C_i'')$;
  - compute $M = C'/e(P,P)^{sk \cdot x}$.

It is not difficult ot check that the decryption is correct if and only if $\sum_{i \in L} f(i) = sk$.

Attribute based encryption

- Key-policy attribute based encryption (KP-ABE): access policy tree is stored in the decryption key;
- Ciphertext-policy attribute based encryption (CP-ABE): access policy tree is stored in the ciphertext.

3.6 Blockchain and cryptocurrency

From wikipedia, *A blockchain – originally block chain – is a distributed database that maintains a continuously growing list of records, called blocks, secured from tampering and revision. [...] blockchain database is managed autonomously.* The motivation is that people want to get rid of central authorities holding (private) states. People can use distributed databases with rules we all agree on (as a society), but there are some direct problems:

- others have no clue about one's balance;
- bitcoins might be double spent.

The idea is to broadcast transaction and the majority can prevent invalid spendings, but further problems comes since it's anonymous network and everyone can spawn as many identities as he / she wishes (sibling attack). If a group is in possession of $> 50\%$ of the networks entity, trust is lost. Things are however changed back in 2008.

Overview of blockchian

1. Transaction: two Parties exchange data; this could represent money, contracts, deeds medical record, customer details, or any other asset that can be described in digital form. A transaction is an *executable script*:

    - its language could be turing complete;
    - in case of Bitcoin, it has the form: input - transaction(s) to be spent; output - receiver(s).

2. Verification: depending on the network's parameters, the transaction is either verified instantly or transcribed into a secured record and placed in a queue of pending transactions. In this case, node – the computers or servers in the network – determine if all transactions are valid based on a set of rules the network has agreed to. In blockchain, the gradients are

    - valid "wallet" address, which is the hash of the public key (in bitcoin, it is HASH256 of ECDSA public key);
    - transaction signed with ECDSA;
    - input-value equalling output value;
    - input-transaction(s) are valid.

3. Structure: each block is identified by a 256-bit hash number, created using algorithm agreed upon by the network. A block contains a header, a reference to the previous block's hash, a nonce to randomize the hash output and a group of transactions represented by a Merkle tree (Fig. 3.4). The sequence of linked hashes creates a secure, interdependent chain.

$$\text{BlockHash} = H(\text{prevBlockHash}||\text{MerkleTop}||\text{Nonce}). \tag{3.12}$$

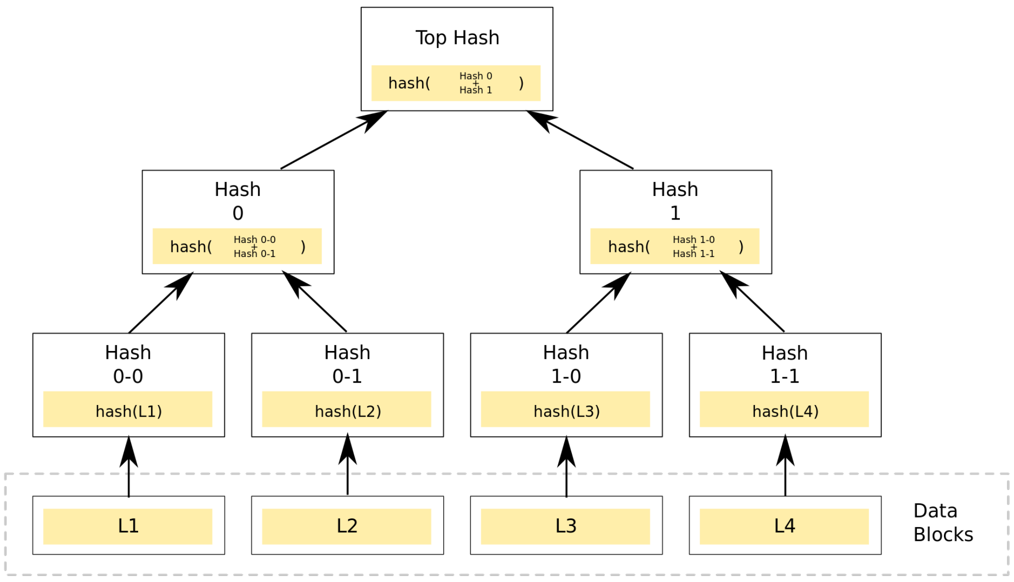


Figure 3.4: Merkle tree.

4. Validation: blocks must first be validated to be added to the blockchain. The most accepted form of validation for open-source blockchains is *proof of work* - the solution is a mathematical puzzle derived from the block's header.

5. Blockchain mining: miners try to "solve" the block by making incremental changes to one variable until the solution satisfies a network-wide target. This is called "proof of work" because correct answers cannot be falsified; potential solutions must prove the appropriate level of computing power was drained in solving.

   For instance, to solve a hash puzzle is to find the nonce of the given block such that

$$H(\text{prevBlockHash}||\text{MerkleTop}||\text{Nonce}) \leq t. \tag{3.13}$$

   The target t is chosen such that the network requires $\sim$ 10 minutes to solve the puzzle. The current hash rate is $\sim 1.4 \times 10^{20}$ hashes per second, which is $\sim 2^{76}$ hashes per 10 minutes. Therefore we want $P(h \leq t) \sim 2^{-76}$ and the current Bitcoin-target is 0x170FFEDD, where 0x17 the length of target in bytes and 0x0FFEDD is the value of the target.

6. The Chain: when a block is validated, the miners that solved the puzzle are rewarded and the block is distributed through the network. Each node adds the block to the majority chain, the network's immutable and auditable blockchain.
7. Built-In Defense: if a malicious miner tries to submit an altered block to the chain, the hash function of that block, and all following blocks, would change. The other nodes would detect these changes and reject the blocks from the majority chain, preventing corruption.

Notes

- Each block holds the hash of the previous block and changing a single transaction in a previous block makes all following blocks invalid.
- Only the longest chain in trusted in the network.
- A blockchain is tamper evident and tamper resistant.
- A blockchain is not *immutable*!

Validation The proof of work is energy costing: bitcoin devours more electricity than Switzerland! Some alternatives are

- proof of stake, where the likelihood of a user to publish a new block is tied to the ratio of their stake to the overall blockchain network, which makes the richer get richer;
- delegated Proof of Stake: instead of using the stake to create a block, the user's use their stack to vote for a miner;
- proof of Space: similar to proof of work, but here the puzzle favors the available disk space instead of the CPU / GPU capacity.But the disk space is as endless as computing capacities;
- proof of burn: the user that ”burns” most money in one round, is allowed to publish the new block; all burned money is lost. This also makes the rich get richer;
- proof of elapsed time: a secure hardware time source generates a random wait time. Once the first node wakes up, it creates and publishes a block to the network, alerting the other nodes of the new block; any publishing node stop waiting. The (central) time source must be trusted, and it is no longer distributed;

- proof of authority / identity: only authorized user's are allowed to participate in the voting process. Again, it is no more distributed, but (de-)centralized trust.

In summarize, the solutions either require a vast amount of (natural) recourses, favor "rich" members or undermine the distributed approach and result in a (de-)centralized database.

The 51% **attack** We assume the attacker controls more than 50% of mining nodes in the network (e.g. "hash rate"). The higher the hash rate, the higher the probability to approve your own blockchain, even if it is tampered. Given p (probability of an honest node finds next block), z (approved blocks linked after the tampered block) and $\lambda = z\frac{1-p}{p}$ (attackers potential poisson during approval), the probability of approving a blockchain can be calculated by

$$P = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - \frac{1-p}{p}\right)^{z-k}, \tag{3.14}$$

which always touches 1 when $p \to \frac{1}{2}^+$, so this attack always works! Bitcoin requires $z = 6$, which is enough for the current pool distribution (largest one $\sim 21\%$).

3.7 Network security

What is informtaion security?

Information security ... preservation of confidentiality (2.13), integrity (2.36) and availability (2.10) of information ... NOTE In addition, other properties, such as authenticity (2.9), accountability (2.2), non-repudiation (2.49), and reliability (2.56) can also be involved.
([ISO/IEC27000, 2.30])

- Confidentiality: property that information is not made available or disclosed to unauthorized individuals, entities, or processes. *Eve cannot read the information that Alice sent to Bob.*

- Integrity: property of protecting the accuracy and completeness of assets. *Eve cannot manipulate (in an unnoticed way) the information that Alice sent to Bob.*

- Availability: property of being accessible and usable upon demand by an authorized entity. *Since Alice is authorized to access information, she is granted access. Neither Bob nor Eve have access to the information, if not authorized.*

- Authenticity: property that an entity is what it claims to be. *Alice authenticates with Bob. Bob authenticates the identity of Alice, and thus proves the authenticity of her message. Eve cannot be authenticated as Alice by Bob.*

- Accountability: assignment of actions and decisions to an entity. *Alice is accountable for the information presented. She controls and monitors all access and is responsible for the correctness and security of the information.*

- Non-repudiation: ability to prove the occurrence of a claimed event or action and its originating entities. *Alice cannot deny having sent the message to Bob. Bob cannot deny to have received the message from Alice.*

- Reliability: property of consistent intended behavior and results. *Any request of Alice to Bob should result in consistent results, even if Eve disturbs the process.*

How can we achieve information security?

Goal	Cryptographic approaches	
Confidentiality	encryption	homomorphic encryption attribute based encryption
Integrity	Hash	
Availability	signatures identification	attribute based encryption
Authenticity	DSig, MACs	
Accountability	DSig, MACs identification	blockchains
Non-Repudiation	DSig	blockchains
Reliability	bit commitment, ZKP	
State of information	stored / in transit	being processed

Key distribution Recall in RSA, the verifier needs the signer's public key. How to make sure of it?

Theorem 3.15. ***(Boyd)*** *Assuming the absence of a secure channel, two entities cannot establish an authenticated session without the existence of an entity that can mediate between the two and which both parties trust and have a secure channel with.*

- Manually distributing key. The idea is to manually make sure that the other peer has access to the verification key, for instance, personal hand-over of keys, sending public keys to the communication peers or uploading them on a personal server. However, the availability of keys is not always given; manual distribution does not scale very well; it is also difficult to decide trustworthiness. *Real world examples: pretty good privacy (PGP), signal messenger, threema, ...*

- Key distribution center (KDC). The idea to generate keys by some trusted entity. Central point of trust can however cause central point of failure; also some protocols support decentralization.

- Public key infrastructure (PKI). Some trusted party *signs trusted public keys* and generates certificates which can be sent alongside the public key. The certificate authority (CA) *verifies the identity* of the public key owner. By trusting the CA, one can trust all its users; certificate trees and chains allow trusting users of other CAs. The problem is that keys cannot expire, so the certificates need to keep track of expiration / revocation. *Examples: X.509 (web browsing, TLS, ...), DNSSec (naming), S/Mime (E-mail).*

- Web of trust. The idea of PKI is extended, but the role of (hierarchical) CAs is replaced by a distributed trust mechanism. That is, user signs other user's certificate. The more users sign a key, the more trustworthy it is. The problem is to require enough trustworthy users and incentives to participate in the system. *Example: PGP.*

- Hybrid schemes. Public key cryptography requires large keys and is less efficient. The idea is to use a key agreement protocol to establish symmetric keys. But we need to take care of the man in the middle attack!

Security parameters of key exchange protocols

- Secrecy: If a session is established between two honest peers then no third party should be able to learn any information about the resultant session key (in particular, no such third party, watching or interfering with the protocol run, should be able to distinguish the session key from a random key).
- Authentication: Each party of a Key Establishment (referred to as a session) needs to be able to uniquely verify the identity of the peer with which the session key is exchanged.
- Consistency: If two honest parties establish a common session key then both need to have a consistent view of who the peers to the session are. Namely, if a party A establishes a key K and believes the peer to the exchange to be B, then if B establishes the session key K, it needs to believe that the peer to the exchange is A; and vice-versa.
- (Perfect) Forward secrecy: An attacker who compromises a long-term secret after a completed session should not be able to learn anything about the short-term keys derived in that session.
- Identity hiding: A passive adversary should not learn the identity of partners to a session.
- Aliveness: If A completes a session with peer B, then A has a proof that B was “alive” during the execution of the protocol (e.g., by obtaining B’s unique authentication on some nonce freshly generated by A).
- Peer Awareness: When A completes a session with peer B, A has a guarantee that (not only is B alive but) B has initiated a corresponding session with peer A.

Attacker model (Dolev-Yao, 1983) Assume Eve can

- obtain any message passing through the network (passive);
- be a legitimate user of the network, and thus in particular can initiate a conversation with any other user;
- have the opportunity to be a receiver to any user A. More generally, we allow the possibility that any user B may become a receiver to any other user A.

This matches how we modeled Eve in the context of RSA/DH.

- Secrecy: We perform the DH key exchange to prevent the man-in-the-middle attack.
- Authenticity: Use asymmetric identification with random challenges.
- Consistency:
  - The idea is to include something in the signature that Eve does not know, for instance, use g^{ab} in RSA to encrypt the signature.
  - If Eve can register P_A as her key at a CA that is trusted by Bob, it means the CA does not require to prove the ownership of a corresponding private key (*proof of possession*). It mostly shows that encryption is the wrong cryptographic technique to prove ownership of a key. The idea it to include the identity in the signature that proves the possession of g^{ab}. Note that it only works if the signature is generated with the identity of the verifier instead of the signer.
- (Perfect Forward Secrecy): use the ”long-term-secret” (e.g., RSA private key) only for signing, but not for key transportation, and use fresh DH for every new session (motivation for TLS 1.2 $\rightarrow$ TLS 1.3). One can also apply a key derivation function that provides forward security as long as the DH key is secret.
- Identity hiding: encrypt the identities to protect Alice from active attackers and Bob from passive attackers.
- Aliveness and Peer Awareness: acknowledge that teh session is established.

The SIGMA protocol family that shows the above properties is proven secure and serves as the basis for many modern security protocols: IKEv2, TLS 1.3, Signal Messenger, etc. Modern protocols are formally verified against (subsets) of the presented properties. This can be done by Automated Theorem Provers, e.g. ProVerif or Tamarin.

Part II

Quantum Physics in Cryptography

Chapter 4

Post quantum cryptography

4.1 Quantum computing

See notes on quantum information for details.

4.1.1 Basics

	Classical computer	Quantum computer
Bit set (register)	$\{0, 1\}$	Qubits
Gates	AND, OR, XOR, NOT	Unitary operations
Circuit	Combination of gates	

Examples of quantum gates

- Pauli gates: $\sigma^{x,y,z}$;
- Hadamard gate: $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$;
- CNOT gate: $\mathbb{1} \oplus \sigma^x$ ($\oplus$: direct sum for matrices, XOR for bits);
- Toffoli gate: $|x_0, x_1, x_2\rangle \to |x_0 \oplus (x_1 \wedge x_2), x_1, x_2\rangle$.

4.1.2 Grover's algorithm

Task $f : \{0, 1\}^n \to \{0, 1\}$. Find $x \in \{0, 1\}^n$ such that $f(x) = 1$.

Gates $O_f : |x\rangle \to (-1)^{f(x)} |x\rangle$, $O_0 : |x\rangle \to (-1)^{\delta_{x,0}} |x\rangle$.

algorithm

- Start from $|\psi_0\rangle = |\omega\rangle = H^{\otimes n}|0\rangle$;
- $G = -H^{\otimes n}O_0H^{\otimes n}O_f = -O_\omega O_f$;
- $|\psi_{k+1}\rangle = G|\psi_k\rangle = -O_\omega O_f|\psi_k\rangle$.
- Note that $|psi_k\rangle = \sin((2k+1)\phi)|\alpha\rangle + \cos((2k+1)\phi)|\beta\rangle$, where $|\alpha\rangle$ lies in the solution space and $|\beta\rangle$ is orthogonal to the solution space. $\sin(\phi) = \sqrt{N_0/2^n}$, where $N_0 \ll 2^n$ is the number of solutions. so for large N, $k = \frac{\pi}{4}\sqrt{\frac{2^n}{N_0}}$ calls are needed, which is a quadratic acceleration.

With Grover's algorithm, we need to double the key / hash size of symmetric cryptography for the same level of security!

4.1.3 Shor's algorithm

Quantum Fourier transform $\Rightarrow$ period finding $\Rightarrow$ Factoring. Discrete logarithm, factoring and Diffie-Hellman are broken!

4.1.4 Are we safe?

Mosca's inequality (Michele Mosca, Professor at University of Waterloo) Let Z be the time until quantum computers break current cryptos, X be the time for migration to new cryptos and Y be the time that data should be secret. We are safe if and only if $X + Y \leq Z$.
What is

- X? We have to find quantum-resistant primitives, research them and adapt IT-infrastructure. Judging by past events, this can take *ages*! (MD5 as example, developed in 1993 and first collision found in 1995. Though highly discouraged in 2005, still used a lot today in 2020.)
- Y? That depends. Nation, companies etc. will be interested in long term secrecy, i.e., 25+ years.
- Z? Expectation: break RSA-2048 in ~ 20 years.

4.2 Lattices

Let $\{ b_i \}_{i=1,\cdots,n} \in \mathbb{R}^n$ be linearly independent and

$$\mathcal{L} = \{ \sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z} \} \tag{4.1}$$

be a lattice. The basis is "*good*" when they are almost orthogonal and "*bad*" when some are almost parallel. Two lattice problems are

- Closest vector problem. Encrypt the message by calculating a point close to the message vector. It is hard for a bad basis but easy with a good one.
- Learning with error problem (LWE).
  - Bob's private key is $S \in \mathbb{Z}_N^n$, while his public keys are $A \in \mathbb{Z}_N^{m \times n}$ and $P \in \mathbb{Z}_N^m$. P is calculated by $P = AS + E$, where $E \in \mathbb{Z}_N$ are small errors. A is overdefined, i.e., $m > n$.
  - To encrypt, Alice lets s_i be the i-th bit of her message s. For each i, Alice randomly adds up some rows of A (each row with probability 1/2) to $u_i \in \mathbb{Z}_N^n$ and P to c_i. If $s_i = 0$, she adds a small error to c_i and otherwise adds a large error.
  - To decrypt, Bob can calculate $c_i - \langle u_i, S \rangle$. Small error - 0; large error - 1.

  Attackers needs to solve the closest vector problem!
- Shortest vector problem (SVP). Note that bad bases don't consist of the shortest vectors. The best known algorithm (Lenstra-Lenstra-Lovász, 1982) runs in polynomial time, but yields only exponential approximation.

As of today, we don't know any polynomial time algorithm that approximates the lattice problems to within polynomial factors. The advantages of lattice-based cryptography include

- good security proofs. We know something about the worst case.
- relatively efficient in practice.
- many known applications including encryption, signatures, key exchange, hashing and even IBC, OT, ZKP.
- well-understood concepts.

4.3 Error correcting codes

Definition 4.1. *($(n,k,d)-$**code**) A linear $(n,k,d)-$code C over a finite field $\mathbb{F}$ is a k-dimensional subspace of $\mathbb{F}^n$ where every two elements differ in at least d bits (dist$(x,y) \geq d$ for $x \neq y$).*

It encodes a k-bit word in n-bit code and can correct up to $\lfloor \frac{d-1}{2} \rfloor$ errors.

Definition 4.2. *A matrix $G \in \mathbb{F}^{k \times n}$ is a **generator matrix** for a $(n,k,d)-$ code C if $C = \langle G \rangle$, i.e., the rows of G span C.*

For a message $m \in \mathbb{F}^k$, it's encoded linearly as $c = mG$.

McEliece encryption sheme (1978)

- The good code matrices G are rare, which can be used as Bob's private keys. Here "good" implies a efficient algorithm to do error correction. It can be multiplied by some hiding matrices to be bad matrices $\hat{G} = PGS$ (public keys). Note that P is a permutation matrix and S is non-singular.

- To encrypt, Alice use the bad matrix and add random errors on purpose: $\hat{c} = \hat{G}m + \hat{e}$

- To decrypt, Bob decodes $c = P^{-1}\hat{c} = GSm + P^{-1}\hat{e}$ to GSm and do matrix inverse to get m.

General decoding problem Given $x \in \mathbb{F}^n$, find $c \in C$ to minimize dist(x,c). If $x = c + e$ and e is a vector of weight at most d, i.e., only a correctable numbers of errors are introduced, then c in uniquely determined. The general decoding problem is NP-hard. The (dis-)advantages of code cryptography include

- good security proofs.

- very big keys. For the most trustworthy version of McEliece, the size is $\sim 1\,\text{MB}$.

- many known applications including encryption, signatures, key exchange, etc.

- very well-understood concepts and very mature crypto.

4.4 Multivariate Systems

Definition 4.3. *Let $f_1, \cdots, f_n$ be multivariate quadratics in $x \in \mathbb{F}_q^k$ with coefficients in $\mathbb{F}_q$. The* ***multivariate quadratic (MQ) equation problem*** *is to solve the equation system*

$$f_i(x) = 0, \quad i = 1, \cdots, n. \tag{4.2}$$

The MQ problem is NP-hard.

Extended isomorphism of polynomials

- Quadratic trapdoor functions $F : \mathbb{F}_q^k \to \mathbb{F}_q^n$ as defined above. It is chosen to be easily invertible.
- $S \in \mathbb{F}_q^{k \times k}$, $T \in \mathbb{F}_q^{n \times n}$: invertible linear maps.
- Public key: $P = T \circ F \circ S$. It is in general hard to invert.

Definition 4.4. *Let P as defined above. The* ***extended isomorphism of polynomials*** *problem is to find affine maps S and T as well as an easily invertible quadratic map F such that $P = T \circ F \circ S$.*

In general, its complexity is not well known.

Oil and Vinegar (OV) The trapdoor function splits the variables in categories o (oil) and v (vinegar) such that $F \sim v{*}v + v{*}o + v + o + \delta$ (quadratic + linear + constant). Here the distribution into oil and vinegar is the *secret*. Finding a solution for the equation system is easy with this knowledge, but hard without it.

- Bob shares the trapdoor function $F(v, o)$ as the public key.
- He wants to create a signature for a message of Hash h in the way $F(v_h, o_h) = h$. He can choose random v_h and solve the linear equations for o_h. The signature is (v_h, o_h).

The (dis-)advantages of multivariate cryptography include

- security hard to understand (but has aged well).
- quite efficient and well-suited for resource-limited devices.
- many known applications including encryption, signatures, key exchange, identity-based crypto, etc.
- simple concepts.

4.5 Isogenies on supersingular elliptic curves

Definition 4.5. *An* ***isogeny*** *is a homomorphism between two curves where the kernel is finite.*

$j-$invariant is the same for isomorphic elliptic curves.

- Alice and BOb have agreed on an elliptic curve group E and points P_A, P_B, Q_A, Q_B.
- Choose randomly $m_{A,B}, n_{A,B} \in \mathbb{Z}_p$ and $r_{A,B} = m_{A,B}P_{A,B} + n_{A,B}Q_{A,B}$. Let $\phi_{A,B} : E \to E/r_{A,B}$ be isogenies of E.
- Send $\phi_{A,B}(P_{B,A}, Q_{B,A})$ to each other.
- Check $\phi_{AB} : E_{A,B} \to E_{AB} = E_{A,B}/\phi_{B,A}(r_{A,B})$ and compute the j-invariant of E_{AB} as the shared key.

It is hard to calculate the isogeny between two supersingular elliptic curves without knowledge of kernel but easy with it.
The (dis-)advantages of isogeny-based cryptography include

- it is very young (developed in 2011) and there are no serious threats since restricted to supersingular curve.
- computation of keys is notoriously inefficient, though the keys are very small (~ 330 bit).
- many known applications including encryption, signatures, etc. It is known for its Diffie-Hellman like character.

4.6 Hash-based signatures

Winternitz one-time signature

- Alice chooses x_i as her private keys and $h_i = \text{Hash}^{m_i}(x_i)$ as public keys. Here $m_i \in \{1, 2, 3, 4\}$. Both keys are twice as big as the message.
- Alice publish half of her private key as the (one-time) signature. The choice of m_i comes from the $2i - 1$ and $2i$-th bits of the message. Bob accepts if the values match.

Many-time signature scheme: Merkle Tree for XMSS

- The root of a Merkle tree is published as a part of the public key.
- To sign, send one of the public keys and the corresponding signature as well as the authentication path to the root of Merkle tree (XMSS peak).
- To verify, just hash along the path to the root.

The (dis-)advantages of Hash-based cryptography include

- well-understood security relying only on security of hash functions.
- relatively efficient.
- only used for signing.

Chapter 5

Quantum key distribution and everlasting security

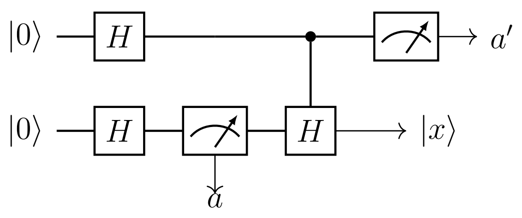
5.1 Quantum key distribution (QKD)

We want information-theoretically secure encryption scheme to avoid quantum attacks, for instance, the one-time pad. But how to exchange enough key-material, especially if every computationally secure key exchange is breakable? The solution is to

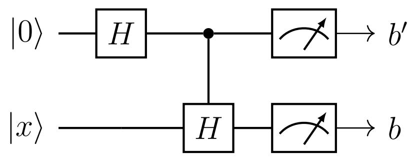
- use quantum computers / effects for key exchange;
- find an information-theoretically secure key exchange.

BB84 protocol.

1. Alice prepare a state $|x\rangle$ and random numbers a, a'.



2. Alice transfer $|x\rangle$ to Bob in a quantum chanel.
3. Bob prepares also two random numbers b, b' using $|x\rangle$.

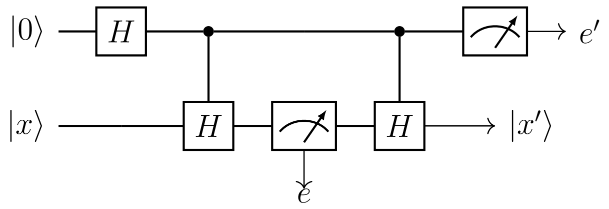


4. Alice and Bob transfers a' and b' to each other in classical channels.
5. If $a' = b'$, Alice chooses $K_i = a$ and Bob chooses $K_i = b$.
6. Repeat until the length of K is equal to the length of message. K can now be used as the key of a one-time pad.

The correctness of BB84 can be seen from the table below:

a'	0	1	0	1	0	1	0	1
a	0	0	1	1	0	0	1	1
$\|x\rangle$	$\|0\rangle$	$H\|0\rangle$	$\|1\rangle$	$H\|1\rangle$	$\|0\rangle$	$H\|0\rangle$	$\|1\rangle$	$H\|1\rangle$
b'	0	0	0	0	1	1	1	1
ctrl(H) $\|x\rangle$	$\|0\rangle$	$H\|0\rangle$	$\|1\rangle$	$H\|1\rangle$	$H\|0\rangle$	$\|0\rangle$	$H\|1\rangle$	$\|1\rangle$
b	0	R	1	R	R	0	R	1

When $a' = b'$, it is always true that $a = b$; otherwise there is only 50% chance. The BB84-protocol allows the detection of *passive* attackers (who can obtain any message passing through the network), for the no-cloning theorem and that the measurements destroy the qubits.
The man-in-the-middle attack is however still possible, if Eve produces e, e' (sent to both Alice and Bob) and $|x'\rangle$ (sent to Bob) as



We already know how to prevent the man-in-the-middle attacks, by using an authentication mechanism to prevent that Eve intercepts the classical chanel. Since Eve cannot read or change a' and b', her probability to correctly guess e' is only $1/2$. Alice and Bob can do k exchanges, and the if $a' = b'$ with $a \neq b$, Eve is detected. The probability *not* to detect Eve is therefore $(3/4)^k$.

Realistic realization Photons can be used as qubits, no entanglement required. We can technically use send light over large distances

- with fibre, with current maximum key rate 17 Mbit s^{-1} (by Toshiba) and maximum distance 404 km.

- in free space, with maximum key rate few kbit s^{-1} and maximum distance $\sim$ 7600 km (Micius satellite).

The challenges however lie in that the noisy signal looks like a man-in-the-middle attack. The distance is therefore given by the max quality of a single link. Historically, amplifying a signal is copying the signal, which again looks like the man-in-the-middle attack. We can use trusted nodes to repeat the signal, but all of them have access to the key and there is no end-to-end security.
QKD can be used in the cases that require information-theoretic security for a single link, e.g., connecting highly classified computing centres. The QKD key rate we can reach now is however much smaller than the link bandwidth. The security advantages of QKD are lost if not used with one time pads.

Pros and Cons of QKD

Pros.

- passive attacks are impossible due to physical laws;
- allows key exchange for information theoretical secure one-time-pads;
- comparable easy quantum circuits, which is already practical today;
- other QKD protocols including E91 (built on entanglement) and E92 are available.

Cons.

- active man-in-the-middle attacks require additional authentication, which also applies for other QKD protocols;
- requires a quantum and a digital channel;
- low key rates;

5.2 Information-theoretically secure authentication

The overall security of the protocol depends on its weakest part, which is the authentication in QKD. People think authentication only needs to be safe in the short term, so currently it is done. But it could happen in the future that the authentication can be broken quickly. To achieve an information-theoretically secure authentication, a good cryptographic hash function is needed.
Let l be the security parameter and use the galois field $GF(2^l)$ with 2^l elements. Consider a message of length n that can be split into n/l blocks, choose a key $k \in GF(2^l)$ and a keyed hash-function $h_k : \{0,1\}^* \to \{0,1\}^l$ is

$$h_k(M) = \sum_{i=1}^{n/l} M_i k^{i-1}. \tag{5.1}$$

If there is collision of h_k such that $h_k(M) = h_k(N)$, $M \neq N$. Then we have

$$h_k(M-N) = 0, \tag{5.2}$$

which can be read as a non-zero polynomial $p_{M-N}(x)$ of degree at most $n/l-1$ with coefficients $M_i - N_i$ evaluated at k. Then k is one of the at most $n/l-1$ roots of the polynomial and the probability that k leads to collision is upper bounded by

$$P(n,l) = \frac{n/l-1}{2^l} \approx \frac{n}{2^l \cdot l}. \tag{5.3}$$

The probability is independent on Eve's computing power and decays exponentially in l.
Ideally authenticated QKD can be continued indefinitely with one pre-shared key for authentication, even though the key the consumed. But in reality exchanging that many qubits all the time is very costly. To conclude, a QKD system is information-theoretically secure only if

- the exchanged key is used in a one-time-pad;
- it is authenticated;
- the authentication mechanism is information-theoretically secure. A pre-shared key is necessary here.

5.3 Everlasting security

Bounded storage model. It is possible that one store the encrypted message for now and crack later, when someone wakes up with a new groundbreaking idea of higher computational power becomes available. Thus the security does not only depend on the computational power of adversaries but also on the maximum storage capacity at given time. The bounded storage model can avoid this strategy.

- Alice and Bob want to encrypt a message M of length m. They share a secret key S of length $k \log n$, where k is the security parameter.
- There is public available random bitstring a of length nm.
- They pick m bitstrings of length k from a and XOR all of these k bits with S to receive a key-bit K_i.

When Eve's available storage is $\lesssim 0.3n$, she cannot store all the random bitstrings.
The potential problem is that the security relies on the mechanism used to exchange the secret key S. Let

- T_A = time to break the key exchange mechanism without knowing the corresponding secrets;
- T_R = time to compute the key exchange of secret key S of length $k \log n$;
- T_N = time to broadcast the random bitstring a of length n.

Even if Eve reveals the secret key S, she cannot reveal the encryption key K is

$$T_A > T_R + mT_N. \tag{5.4}$$

In that case, Eve will retrieve $\leq 30\%$ of the encryption key and the message is secure forever!
Pros of BSM.

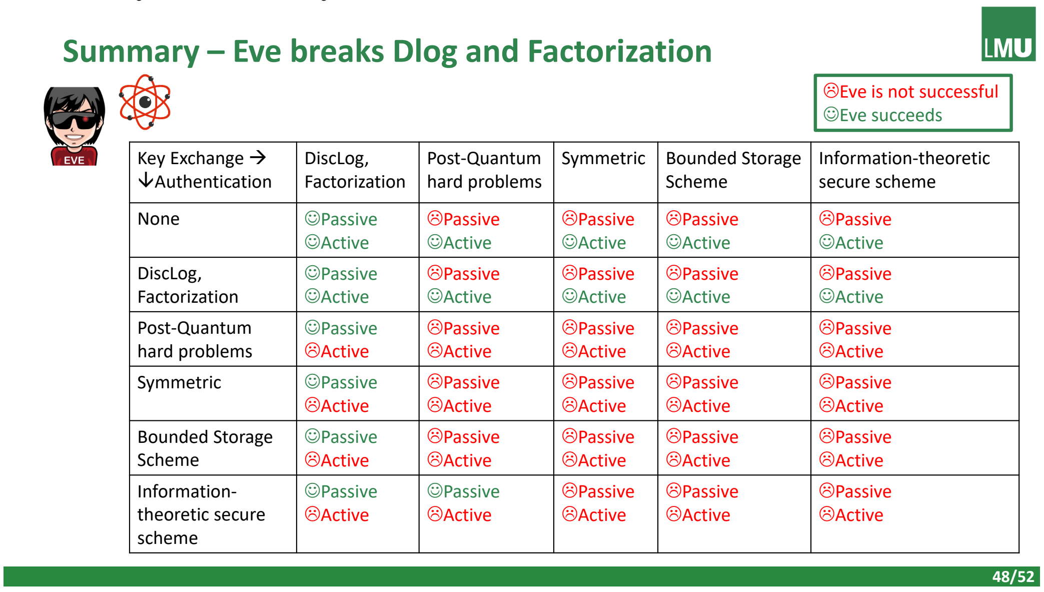
- not limited to encryption;
- current infrastructure like the satellites can be used for broadcasting the random string;
- with some modifications it may be feasible in some setups;

- random string must not be true random, but only claim Kolmogorov complexity.

Cons of BSM.

- no pseudo random number generators can claim Kolmogorov complexity;
- that generating and sending large enough random strings is faster than storing them is required, which is at least currently impracticable.

Summary on security

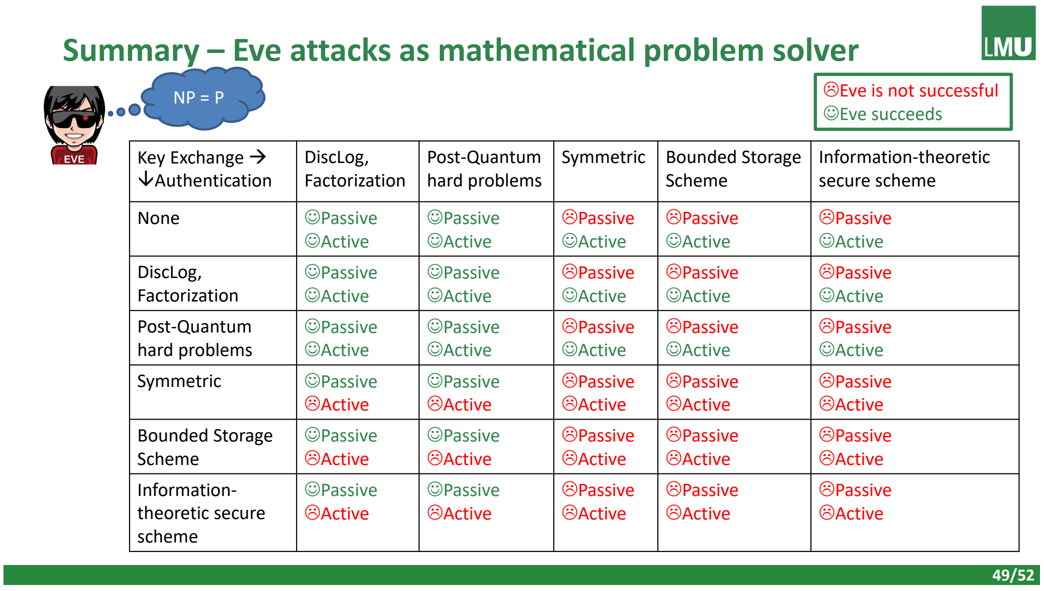


Summary – Eve breaks Dlog and Factorization

☹Eve is not successful
☺Eve succeeds

Key Exchange → ↓Authentication	DiscLog, Factorization	Post-Quantum hard problems	Symmetric	Bounded Storage Scheme	Information-theoretic secure scheme
None	☺Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active
DiscLog, Factorization	☺Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active
Post-Quantum hard problems	☺Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active
Symmetric	☺Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active
Bounded Storage Scheme	☺Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active
Information-theoretic secure scheme	☺Passive ☹Active	☺Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active

48/52

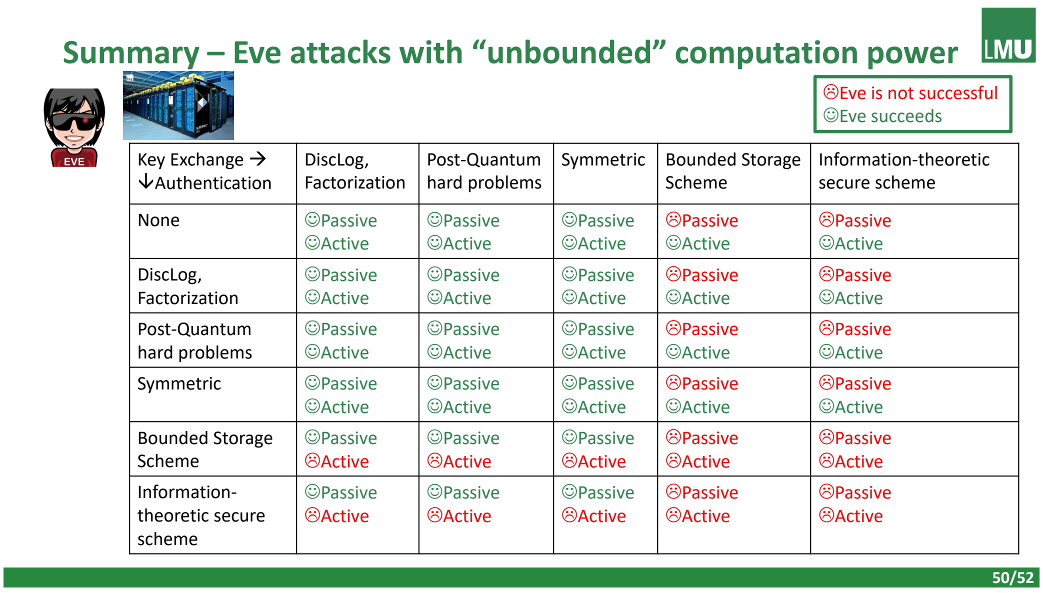


Summary – Eve attacks as mathematical problem solver

☹Eve is not successful
☺Eve succeeds

Key Exchange → ↓Authentication	DiscLog, Factorization	Post-Quantum hard problems	Symmetric	Bounded Storage Scheme	Information-theoretic secure scheme
None	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active
DiscLog, Factorization	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active
Post-Quantum hard problems	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active
Symmetric	☺Passive ☹Active	☺Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active
Bounded Storage Scheme	☺Passive ☹Active	☺Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active
Information-theoretic secure scheme	☺Passive ☹Active	☺Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active

49/52

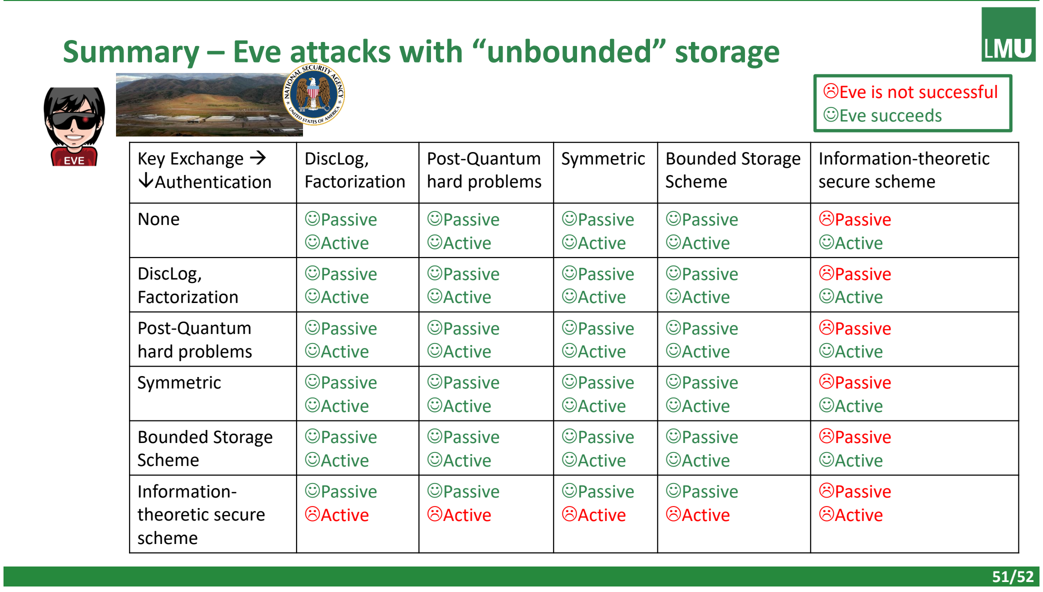


Summary – Eve attacks with “unbounded” computation power

☹Eve is not successful
☺Eve succeeds

Key Exchange → ↓Authentication	DiscLog, Factorization	Post-Quantum hard problems	Symmetric	Bounded Storage Scheme	Information-theoretic secure scheme
None	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active
DiscLog, Factorization	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active
Post-Quantum hard problems	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active
Symmetric	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active	☹Passive ☺Active
Bounded Storage Scheme	☺Passive ☹Active	☺Passive ☹Active	☺Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active
Information-theoretic secure scheme	☺Passive ☹Active	☺Passive ☹Active	☺Passive ☹Active	☹Passive ☹Active	☹Passive ☹Active

50/52



Summary – Eve attacks with “unbounded” storage

☹Eve is not successful
☺Eve succeeds

Key Exchange → ↓Authentication	DiscLog, Factorization	Post-Quantum hard problems	Symmetric	Bounded Storage Scheme	Information-theoretic secure scheme
None	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active
DiscLog, Factorization	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active
Post-Quantum hard problems	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active
Symmetric	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active
Bounded Storage Scheme	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☺Passive ☺Active	☹Passive ☺Active
Information-theoretic secure scheme	☺Passive ☹Active	☺Passive ☹Active	☺Passive ☹Active	☺Passive ☹Active	☹Passive ☹Active

51/52

Part III

Classical Physics in Cryptography

Chapter 6

Hardware security

6.1 Background

Motivation

- Manipulating the hardware components that implement security functions can compromise system integrity, provide unauthorized access to protect data, and endanger intellectual property.
- Secure hardware is required to protect software in a proper manner tampering.
- Addressing the vulnerabilities is essential to prevent the hardware from becoming the Achilles heel of today's systems.
- Massive utilization of hardware circuits in larger cyberphysical systems that are interacting with the physical environment via sensors and actuators.
- Cyberphysical systems are more and more integrated via open networks, most notably the Internet.

CIA triad in hardware security

- Confidentiality: prevent unauthorized disclosure of IP (Intellectual Property, software or hardware) and communication messages from / to the chip.
- Integrity: prevent unauthorized tampering with IP nad communication messages.

- Availability: chips remain operational and useable whenever needed by an authorized entity.

There have been arising hardware security problems because of the global trends in IC design, manufacturing and distribution in the supply chain.

Hardware-based threats

- Hardware Trojans: an attacker either in the design house or in the foundry may add malicious circuits or modify existing circuits to bypass, disable the security fence or destroy the chip.
- IP piracy and IC overbuilding: an IP user or a rogue foundry may illegally pirate the IP without the knowledge and consent of the designer. A malicious foundry may build more than the required number of ICs and seels the excess ICs in the gray market.
- Reverse engineering: an attacker can reverse engineer the IC / IP design to his / her desired abstraction level. He can then reuse the recovered IP or improve it.
- Side-channel analysis: an attacker can extract the secret information or secret keys by exploiting a physical modality (power consumption, timing, or electromagnetic emission) of the hardware that executes the target application.

Real stories

- Backdoors to the ”secure” telex, radio and teletype machines and sell them to 160 countries including Iran, Libya, Yugoslavia. It is done by cooperation between NSA (the US) and BND (Germany), aimed to intercept communication between countries and their embassies.
- Sky and NDS hired hackers to improve the security of their payTV products and hack the competitor’s products.
- In September 2007, Israeli jets bombed a suspected nuclear installation in northeastern Syria. The failure of a Syrian radar, supposedly state of art, was because a commercial off-the-shelf microprocessors in it might have been purposely fabricated with a hidden backdoor inside.
- Boeing blamed a Xilinx FPGA for the failure of an ice-detection module in P-8A Poseidon. Retracting the FPGA’s path led to a Chinese company A access electronics that sells used Xilinx parts as new.

6.2 Physical attacks

Recall the *Kerckhoffs' principle* for symmetric cryptos: a crypto system should be secure even if everything about the system except the key is public knowledge. Modern cryptography algorithms are therefore evaluated and analyzed by the community. They are however mathematical analysis to find the theoretical weakness. Physical attacks are to exploit weakness in the implementation of the cryptographic algorithms.

- Requirements:
  - Direct access to the chip;
  - connection to signals;
  - equipments and knowledge (focus ion beam, photonic emission analysis, ...).
- Interaction: exploiting some physical characteristics of the device.
- Exploitation: analyzing the gathered information to recover the secret.

Attackers

- Class I: clever outsiders. Insufficient knowledge of the system and limited access to the equipments and tools.
- Class II: knowledgeable insiders. Knowledge of the system and access to equipments and tools.
- Class III: founded organizations. Access to all resources.

Attacker motivations

- Direct theft of service of money: smart card, game console, ...
- Sell of products: IP piracy, cloning, overbuilding, counterfeiting.
- Denial of service: competitors' devices.
- In all, breaking the crypto system.

Classification

- Non-invasive: passive; no device damage, no tamper evidence; most low cost and repeatable. Examples:
  - Side-channel attacks: monitor / measure the chip's physical characteristics (power, current, timing, EM radiation, etc.) during its normal operation. Perform data analysis to learn information.
  - software attacks: use normal I/O interface; exploit known security vulnerabilities in protocols, algorithms and their software implementation.
- fully-invasive: direct access to the inside of the chip / device; irreversible; device damaged or tamper evidence left. Examples:
  - Reverse engineering.
  - Micro-probing.
- semi-invasive: access to the surface of the chip, but not internal wires; does not damage the system; may or may not leave temper evidence; moderate cost and some special skills; repeatable - high cost and requirement of skills. E.g., fault generation (could be non-invasive).

Side-channel attacks

In 1943 Bell labs employees working on encryption systems noted that whenever the system is activated, spikes appeared on an oscilloscope in another part of the lab, which could be interpreted to recover the plaintext data. In 1950s British intelligence agents listened to the resetting of the key wheels of Egyptian encryption machine to reduce their starting positions, which enabled them to crack the encryption.
The most famous side-channel attack, the Thing, also known as the Great Seal bug, was one of the fist covert listening devices (or "bugs) to use passive techniques to transmit an audio signal. It was concealed inside a gift given by the Soviet Union to W. Averell Harriman, the US ambassador in 1945. As is needed electromagnetic energy from an outside source to become energized and active, it is considered a predecessor of Radio-Frequency Identification (RFID) technology.
The sources of the side channel includes sound and EM field (including light) emitted, power consumption, timing or delay and output signals. For instance, a change of input might lead to dynamic power.

Chapter 7

Introduction to physical unclonable functions

From the last chapter, we know it difficult to *store* secret, digital cryptographic keys safely and permanently in electronic hardware. Can we

- design security hardware *without the long-term presence* of secret keys in digital form? – Weak physical unclonable functions (weak PUFs).
- design security hardware *without any secret keys at all*? – Strong physical unclonable functions (strong PUFs).

Physical disorder The small-scale structure of almost any mesoscopic and macroscopic object is not perfectly smooth, but random, imperfect, unique or physically disordered. They are very hard to duplicate or to clone, even for their original manufacturers. The technology for perfect duplication in 3D does not exist yet. The disorder is usually considered an unwanted phenomenon and a nuisance. *Can we exploit it constructively, too?*

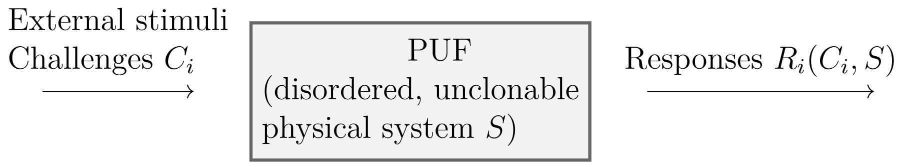


Figure 7.1: PUF setup

(C_i, R_i) are called challenge-response pairs (CRPs) of the PUF. The CRPs are not perfectly but at least halfway stable. The Zoo of PUFs includes: physical obfuscated keys, weak PUFs, controlled PUFs, physical random functions, strong PUFs, public PUFs, SIMPL systems, etc.

7.1 Weak PUFs

In weak PUFs, specific security features are

- very few possible challenges, and often only one single CRP.
- protected, internal CRP-interface within PUF-embedding hardware, *i.e.,* no access from outside.

Some examples of weak PUFs are

- SRAM PUF. The response is an SRAM array (0 or 1 at each site);
- diode-based weak PUF. CRP is $I - V$-curve;
- transistor-based weak PUF;
- buskeeper PUF;
- DRAM-based weak PUF;
- memristor-based weak PUF;
- flash-based weak PUF;
- etc.

Application The main application of weak PUFs is the *secret key source* in hardware (HW) without non-volatile memory (NVM). In the setup phase, manufacturer externally measures (noisy) weak PUF Responses during fabrication and derives (and stores) secret kep K and error-correcting helper data HD from R_i. Later, HW internally measures weak PUF responses and gets HD from somewhere, that does not have to be secret. HW then derives the secret key K from R_i. Eve cannot access weak PUF!

Pros and Cons of weak PUFs as secret key source

Pros:

- allow key storage / HW-individualization without non-volatile memory on board, which costs less;
- keys not present permanently in digital form, but instead hidden deep in analog hardware features. When the system is powered off, all digital keys vanish automatically, which improves the security level.

Cons:

- practically, perfect error correction for noisy PUF-responses is required on chip;
- weak PUFs take more space than classical NVM, including additional error correction module and dependencies between neighboring weak PUFs in array. Several thousand weak PUFs are required for 128-bit key!
- lead to short-term digital secrets and *permanent physical secrets* in HW, exploited in practical and efficient attacks in the past.

Funny access asymmetry Why can Manufacturer access Weak PUFs during fabrication, while Eve cannot access Weak PUFs later? To overcome this asymmetry, we can use weak PUFs and (private key, public key)-pairs. A trusted authority signs the private key with HW's ID with his one master signing key and store it in NVM. Later in the field, HW can check with the master verification key.

PUF-based security capsules (Strong or Weak) optical PUF-capsule with few or many CRPs, measured from inside uses material that is sensitive when being performed or invasively tampered. It is a material science research question.

7.2 Strong PUFs

In weak PUFs, specific security features are

- very many (ideally exponentially many) possible challenges;
- highly complex challenge-response relation such that no machine learning or numeric simulation of responses even if many CRPs are known;
- public access to CRP interface, but no exhaustive read-out of all CRPs.

Some examples of strong PUFs are

- Optical PUF. The historically first strong PUF. Challenges are points and angles of incidence of laser beams and the responses are speckle patterns with thousands of bits. Disorder comes from positions of the glass spheres. It however requires high precision measurements and positioning.

- Integrated electrical strong PUF. Challenge is a 128-bitstring and the response is a single bit. disorder comes from manufacturing variations and runtime delays.

Applications

- Identification / authentication. The central authority (e.g., the bank HQ) holds a secret list of CRPs of PUF S. HD (e.g., bank card) consumes a CRP in each authentication process that will be removed from the secret list. *Pros: no digital secret keys in hardware and avoids standard computational assumptions.*
- Key exchange. Alice chooses random challenges and derives the key form the responses. She then send the PUF to Bob. After Bob confirmed receiving the PUF, Alice sends the challenges so that Bob can also derive the same key. *It avoids standard computational assumptions like factoring / DH, albeit new assumption on the PUF become necessary.*
- Oblivious transfer, bit commitment, secure multi-party computation, etc.
- Strong PUFs are not just a novel security tool, but also a *universal cryptographic primitive.*

Attacks on PUFs: modeling attacks Modeling attacks are a non-physical, digital way of cloning physical unclonable functions. Adversary Eve collects a small fraction of all strong PUF CRPs. She then extrapolates the PUF behavior on all CRPs, e.g., via machine learning techniques. Usually, certain assumption about the internal functionality of the attacked PUF are mode, that improves machine learning performance.
The super-high information content (SHIC) PUFs has provable security (also the only known strong PUF) against machine learning. The core features are huge entropy, intentionally slow tailor-made read-out speed and diodes with extreme rectification ratios. All CRPs are information-theoretically independent. SHIC PUFs have however large area consumption and slow read-out speed.

PUF-capsules and application in IoT The PUF-capsules serve as new, standard protection tools around IoT (internet of things) endpoints, without need of battery. They are small and lightweight, along with high security levels.

- The PUF-capsule can derive a (SK, PK)-pair within it.
- The manufacturer sign it with the (secret) master signing key. The certificate is $\mathrm{DigSig}_{\mathrm{MSK}}$(HW-ID, HW-Functionality, PK), which is also stored inside the capsule.
- The master verification key is public so that the PUF-capsule can be verified later. Any physical change in the reflector will change the (SK, PK)-pair.

Summary

- Physical disorder is not just a nuisance, but a very powerful security resource.
- It enables key-free identification, authentication and tamper protection.

7.3 Protocols with PUFs

7.3.1 Oblivious transfer

Recall: OT

- Input: receiver holds a choice bit b while the sender holds to secretes s_0 and s_1.
- Output: the receiver obtains the secret s_b.
- Security: receiver cannot learn both secrets, while the sender cannot learn the receiver's choice.

OT is a universal cryptographic primitive that allows implementation of a very large number of other cryptographic protocols including bit commitment, zero-knowledge proofs and secure multiparty computation, etc. It is thus often used as a first touchstone for new cryptographic models, and has been successful in bounded storage model, noise-based crypto and *strong PUFs*. (However fails in quantum crypto!)

OT with strong PUFs

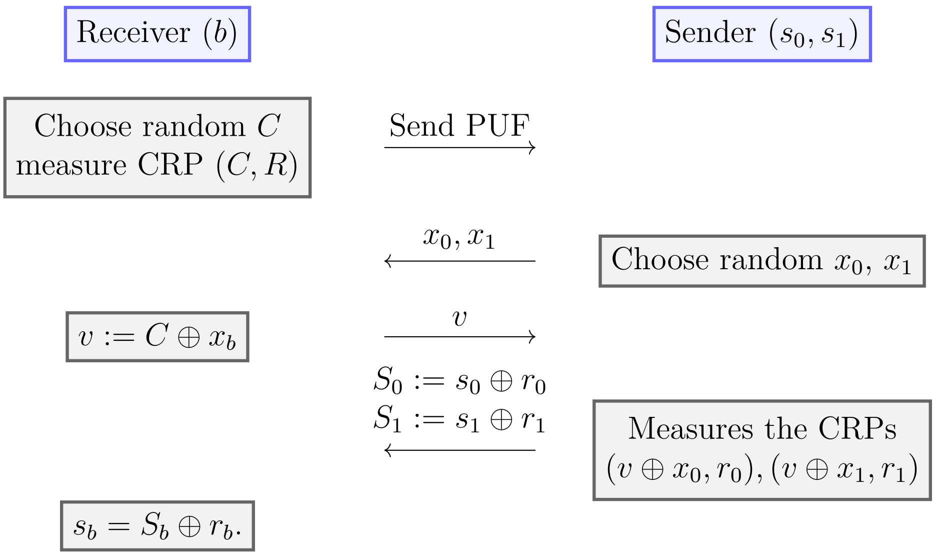


Figure 7.2: OT with strong PUFs.

Note that one PUF can be used only once! Otherwise the receiver could know both secrets. There is currently ongoing research on the practical security features, e.g.,

- what if the sender uses a *malicious* or *bad* PUF?
- quadratic attacks are known on OT-protocol above. In the case of Pappu et al's optical PUF, it can be done within 20 mins.

7.3.2 Everlasting security

Recall A cryptographic protocol is said to possess *everlasting security* if it remains secure even if Eve after the end of the protocol (but not during the protocol!) becomes computationally all-powerful. Protocols with everlasting security include: one-time pad encryption, KE (key exchange) with bounded storage model, QKE, KE / OT with strong PUFs (e.g., SHIC PUFs).
Comparably, SHIC PUFs are the most inexpensive solution. It however requires transfer of physical objects and does not scale well. The speed is also low, taking several todays, though it is the same for bounded storage model.

7.3.3 Secret-free security

Definition 7.1. *A* ***secret*** *is some piece of information present or stored in a hardware H in whatever form (i.e., digital or analog, permanent or volatile,* *positions / states of single atoms or particles, etc.), and whose disclosure to the adversary breaks the security of the considered security scheme S.*

No information is a secret by itself, but only becomes one in a certain context, e.g., relative to a considered security scheme S.

Where can we hope for secret-free security?

- encryption: probably not, as plaintext will always be a secret and present in encryption hardware.
- key exchange: probably not, as exchanged keys will always be a secret and present in hardware after KE protocol.
- identification: yes!
- message authentication: yes!
- *secure, forgery-proof tagging of valuable items*: yes!

International trade in counterfeited/pirated products in 2016: *460, 000, 000, 000 euros (3.3% of world trade)*, not including domestically produced and consumed counterfeit/pirated products, or pirated digital products distributed online.

Definition 7.2. *A physical object O is called a* ***unique object (UNO)*** *with unique properties $P_1, \cdots, P_k$ relative to an external, detachable measurement apparatus M if:*

- *(practicality requirements)*
  - *O has properties $P_1, \cdots, P_k$ upon measurement with M;*
  - *these properties are (relatively) stable, can be measured (relatively) quickly and can be expressed in a (relatively) short bitstring.*
- *(security requirements)*
  - *no other object O' (clone) can be produces with some properties $P_1, \cdots, P_k$ upon analog measurement with external M, not even by the original manufacturer of O, and not even if adversary knows all properties and all internal details of O and M!*
  - *O and M do not contain any secrets, whose disclosure to the adversary breaks security.*

UNO candidates

- optical nano-resonators and their mode spectrum;
- complex radio wave-scattering objects and their scattering behavior;
- strong PUFs and a few CRPs are properties. It can however not be used for secret-free *offline* labels, as there are viagra-fake that use small chips to store (C_i, R_i).

Applications

- passports, ID cards, access cards, bankcards, ...;
- banknotes;
- unclonable data carriers;
- forgery proof tagging of any branded products.

7.4 Virtual proofs of reality

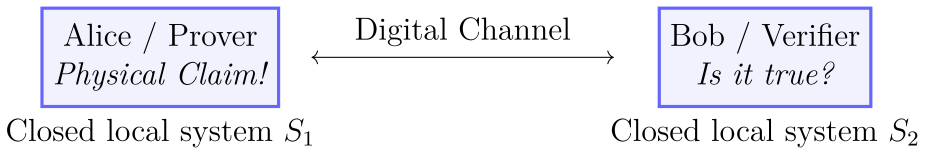


Figure 7.3: Basic idea of virtual proofs (VPs)

- Examples of the claims include: temperature / humidity / pressure in S_1; relative location of two objects in S_1; destruction of a certain object in S_1 within a certain time period; authenticity of video of audio data recorded in S_1; configuration / state of certain hardware in s_1; etc.
- Alice tries to prove a general physical claim regarding s_1 to Bob over the digital channel, *without* using classical secret keys, or classical tamper-proof environments, or classical trusted sensors in S_1.
- It is a natural extension of classical, mathematical crypto and security, in particular of interactive proof systems, into the physical domain. Also classical keys that represent primary attack points can be avoided in the first place.

VP of distance Alice wants to prove to Bob that two objects O_1 and O_2 in S_1 have distance D to each other. The idea is to use two optical PUFs whose interference pattern depends on the unique structure of the PUFs, on their distance and on the challenge C_i. Typical distance resolution is $\sim 100\,\mu\text{m}$.

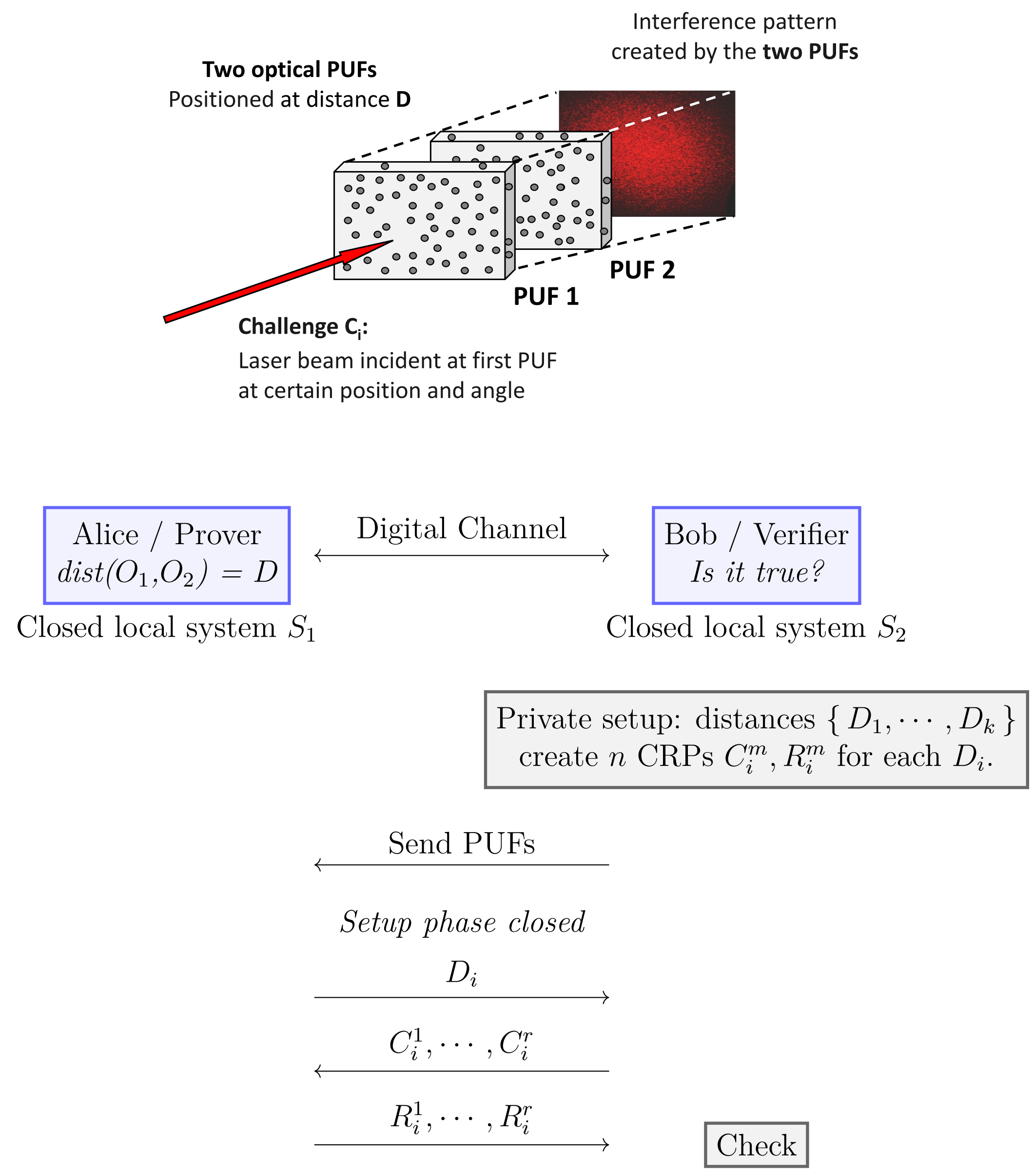

Interference pattern created by the **two PUFs**
Two optical PUFs
Positioned at distance **D**
PUF 2
PUF 1
Challenge C_i:
Laser beam incident at first PUF at certain position and angle
Alice / Prover
$dist(O_1,O_2) = D$
Digital Channel
Bob / Verifier
Is it true?
Closed local system S_1
Closed local system S_2
Private setup: distances $\{D_1, \cdots, D_k\}$ create n CRPs C_i^m, R_i^m for each D_i.
Send PUFs
Setup phase closed
D_i
$C_i^1, \cdots, C_i^r$
$R_i^1, \cdots, R_i^r$
Check

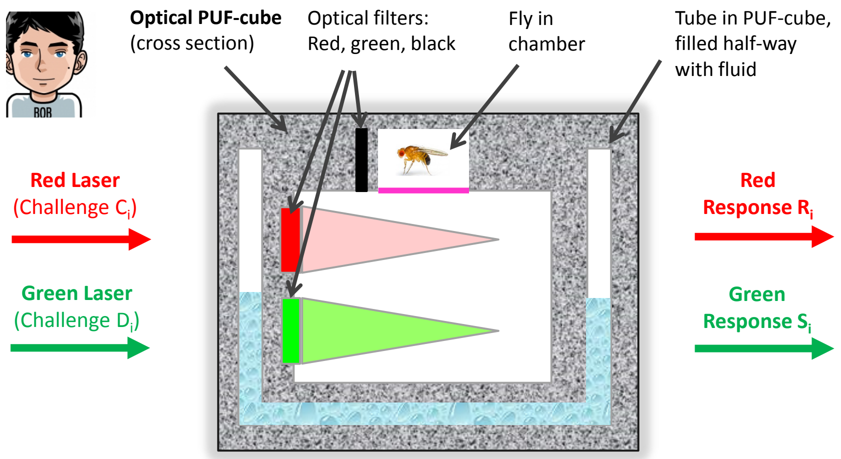
Figure 7.4: VP of distance

VP of temperature Alice wants to prove to Bob that a certain object O is at temperature T at the time of execution of the VP. The idea is to use temperature-sensitive strong PUF, so-called *Bistable ring PUF* as object O. Typical resolution is $\sim 4\,\mathrm{K}$. It can be generalized easily to any other one-dimensional physical variables given that such PUFs can be constructed.

VP of destruction Alice wants to prove that a certain, unambiguously identified object O has been destroyed, or irreversibly modified during the VP. There are two implementations:

- using an optical *PUF inside a PUF*: in order to record challenge-response-pairs from the inner PUF, one must remove and thus destroy the outer PUF first.
- using a quantum protocol: measurement amounts to dstruction of certain quantum states.

VP of life Alice wants to prove that some being / animal, for instance a fly, is alive. The basic idea is to lock the fly in a PUF-cube. If the fly is alive, it will move occasionally and otherwise not moving. The resulting interference patterns are different.


BOB
Optical PUF-cube (cross section)
Optical filters: Red, green, black
Fly in chamber
Tube in PUF-cube, filled half-way with fluid
Red Laser (Challenge C_i)
Green Laser (Challenge D_i)
Red Response R_i
Green Response S_i

- Bob builds the unique and unclonable PUF-cube himself. In the setup phase, he records the challenges with red and green lasers when the fly is in the chamber.
- After the fly is released, it will affect the challenges of red / green laser if it stays in the corresponding shadow.

- If the fly is dead, it could not move from the green shadow to the red shadow. Artificial external movement would also move the fluid and thus change optical response.
- If Alice at times sends correct CRPs from both red and green shadow, then the fly can be believed to still move autonomously.