# A Crush Course On Cryptography

Yu Zhang

Harbin Institute of Technology

Cryptography, Autumn, 2018

# What cryptography is and is not

Cryptography is:

- A tremendous tool
- The basis for many security mechanisms
- Secure communication:
  - web traffic: HTTPS (SSL/TLS)
  - wireless traffic: 802.11i WPA2 (and WEP), GSM, Bluetooth
  - encrypting files on disk: EFS, TrueCrypt
  - content protection: DVD (CSS), Blu-ray (AACS)
  - user authentication

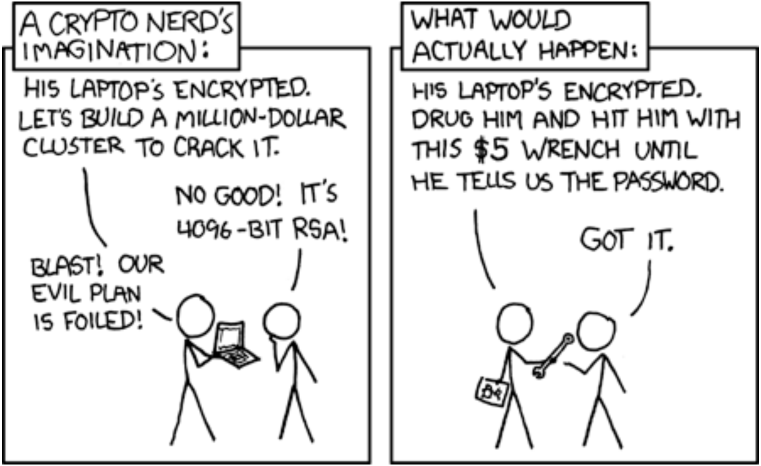
Cryptography is **NOT**:

- The solution to all security problems
- Reliable unless implemented and used properly
- Something you should try to invent yourself

# What cryptography can and can't do

"No one can guarantee 100% security. But we can work toward 100% risk acceptance. ... Strong cryptography can withstand targeted attacks up to a point–the point at which it becomes easier to get the information some other way. ... The good news about cryptography is that we already have the algorithms and protocols we need to secure our systems. The bad news is that that was the easy part; implementing the protocols successfully requires considerable expertise. ... Security is different from any other design requirement, because functionality does not equal quality."

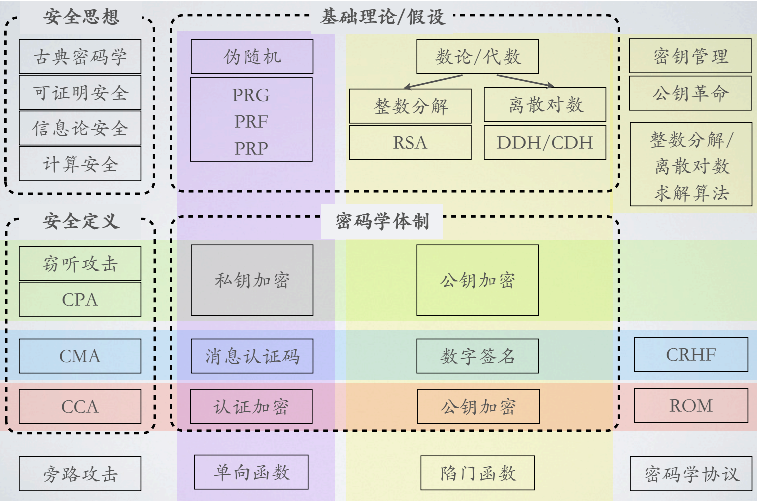
– By Bruce Schneier 1997


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

# Outline

- Classic cryptography, Perfect Secrets
- Private Key Encryption, MAC, Block Cipher, OWF
- Number Theory, Factoring and Discrete Log
- Key Management, Public Key, Digital Signature
- TPD, Random Oracle Model
- Cryptographic Protocols (Many magics here)

# Syllabus [in Chinese]



# We will learn from Turing Award recipients

- 1995 M. Blum
- 2000 A. Yao
- 2002 R. Rivest, A. Shamir, L. Adleman
- 2012 S. Micali, S. Goldwasser
- 2013 L. Lamport
- 2015 M. E. Hellman, W. Diffie

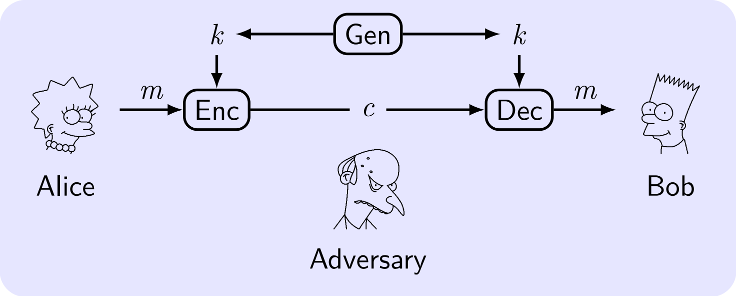
# Securing Key vs Obscuring Algorithm

- Easier to maintain secrecy of a short key
- In case the key is exposed, easier for the honest parties to change the key
- In case many pairs of people, easier to use the same algorithm, but different keys

**Kerckhoffs's principle**

*The cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.*

# The Syntax of Encryption




- key $k \in \mathcal{K}$, plaintext (or message) $m \in \mathcal{M}$, ciphertext $c \in \mathcal{C}$
- **Key-generation** algorithm $k \leftarrow \mathsf{Gen}$
- **Encryption** algorithm $c := \mathsf{Enc}_k(m)$
- **Decryption** algorithm $m := \mathsf{Dec}_k(c)$
- **Encryption scheme**: $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$
- **Basic correctness requirement**: $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$

# One-Time Pad (Vernam's Cipher)

- $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^\ell$.
- Gen chooses a $k$ randomly with probability exactly $2^{-\ell}$.
- $c := \mathsf{Enc}_k(m) = k \oplus m$.
- $m := \mathsf{Dec}_k(c) = k \oplus c$.

**Theorem 1**

*The one-time pad encryption scheme is perfectly-secret.*

# Definition of 'Perfect Secrecy'

**Intuition**: An adversary knows the probability distribution over $\mathcal{M}$. $c$ should have no effect on the knowledge of the adversary; the a *posteriori* likelihood that some $m$ was sent should be no different from the a *priori* probability that $m$ would be sent.

**Definition 2**

$\Pi$ over $\mathcal{M}$ is **perfectly secret** if for every probability distribution over $\mathcal{M}$, $\forall m \in \mathcal{M}$ and $\forall c \in \mathcal{C}$ for which $\Pr[C = c] > 0$:

$$\Pr[M = m | C = c] = \Pr[M = m].$$

**Simplify**: non-zero probabilities for $\forall m \in \mathcal{M}$ and $\forall c \in \mathcal{C}$.

**Is the below scheme perfectly secret?**

For $\mathcal{M} = \mathcal{K} = \{0, 1\}, \mathsf{Enc}_k(m) = m \oplus k$.

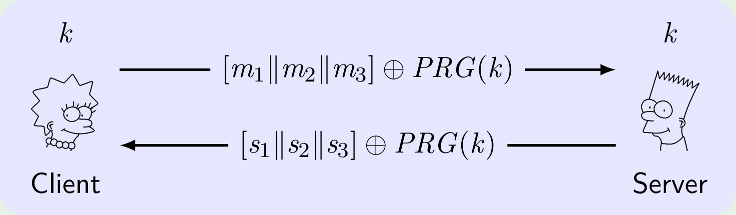
# Two Time Pad: Real World Cases

Only used once for the same key, otherwise

$$c \oplus c' = (m \oplus k) \oplus (m' \oplus k) = m \oplus m'.$$

Learn $m$ from $m \oplus m'$ due to the redundancy of language.

**MS-PPTP (Win NT)**

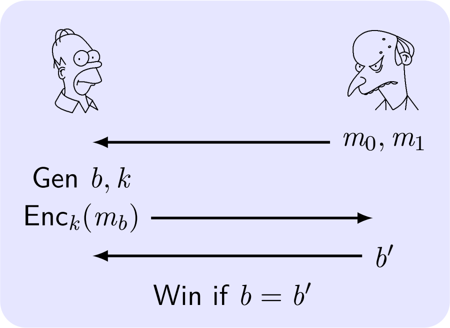



Improvement: use two keys for C-to-S and S-to-C separately.

# Eavesdropping Indistinguishability Experiment

The eavesdropping indistinguishability experiment $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n)$:

1. $\mathcal{A}$ is given input $1^n$, outputs $m_0, m_1$ of the same length
2. $k \leftarrow \mathsf{Gen}(1^n)$, a random bit $b \leftarrow \{0,1\}$ is chosen. Then $c \leftarrow \mathsf{Enc}_k(m_b)$ (challenge ciphertext) is given to $\mathcal{A}$
3. $\mathcal{A}$ outputs $b'$. If $b' = b$, $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi} = 1$, otherwise 0



# Defining Private-key Encryption Security

**Definition 3**

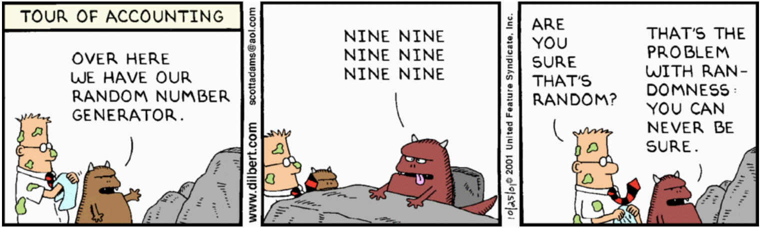
$\Pi$ has **indistinguishable encryptions in the presence of an eavesdropper** if $\forall$ PPT $\mathcal{A}$, $\exists$ a negligible function negl such that

$$\Pr\left[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n),$$

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

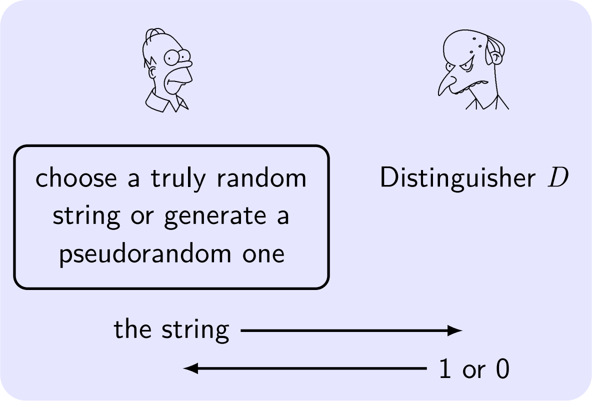
# Conceptual Points of Pseudorandomness

- True randomness can not be generated by a describable mechanism
- Pseudorandom looks truly random for the observers who don't know the mechanism
- No fixed string can be "pseudorandom" which refers to a distribution
- Q: is it possible to definitively prove randomness?



# Intuition for Defining Pseudorandom

**Intuition**: Generate a long string from a short truly random seed, and the pseudorandom string is indistinguishable from truly random strings.



# Definition of Pseudorandom Generators

## Definition 4

A deterministic polynomial-time algorithm $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ is a **pseudorandom generator (PRG)** if

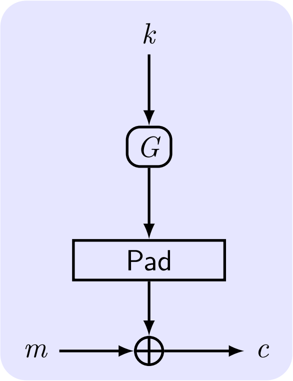
1. (Expansion:) $\forall n, \ell(n) > n$.
2. (Pseudorandomness): $\forall$ PPT distinguishers $D$,

$$|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \leq \mathsf{negl}(n),$$

where $r$ is chosen *u.a.r* from $\{0,1\}^{\ell(n)}$, the **seed** $s$ is chosen *u.a.r* from $\{0,1\}^n$. $\ell(\cdot)$ is the **expansion factor** of $G$.

- Pseudorandomness means being **next-bit unpredictable**, $G$ passes all next bit tests $\iff$ $G$ passes all statistical tests.
- **Existence**: Under the weak assumption that *one-way functions* exists, or $\mathcal{P} \neq \mathcal{NP}$

# A Secure Fixed-Length Encryption Scheme



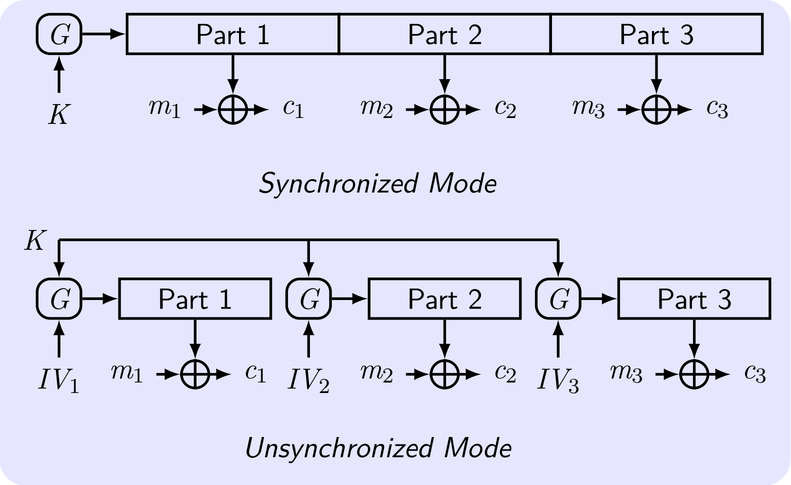

**Construction 5**

- $|G(k)| = \ell(|k|)$, $m \in \{0,1\}^{\ell(n)}$.
- Gen: $k \in \{0,1\}^n$.
- Enc: $c := G(k) \oplus m$.
- Dec: $m := G(k) \oplus c$.

**Theorem 6**

*This fixed-length encryption scheme has indistinguishable encryptions in the presence of an eavesdropper.*

# Secure Multiple Encryptions Using a Stream Cipher




Initial vector $IV$ is chosen *u.a.r* and public

Q: which mode is better in your opinion?

# Related Keys: Real World Cases

Keys (the $IV$-key pair) for multiple enc. must be independent

**Attacks on 802.11b WEP**

Unsynchronized mode: $\mathsf{Enc}(m_i) := \langle IV_i, G(IV_i \| k) \oplus m_i \rangle$

- Length of $IV$ is 24 bits, repeat $IV$ after $2^{24} \approx$ 16M frames
- On some WiFi cards, $IV$ resets to $0$ after power cycle
- $IV_i = IV_{i-1} + 1$. For RC4, recover $k$ after 40,000 frames

# Chosen-Plaintext Attacks (CPA)

**CPA**: the adversary has the ability to obtain the encryption of plaintexts of its choice

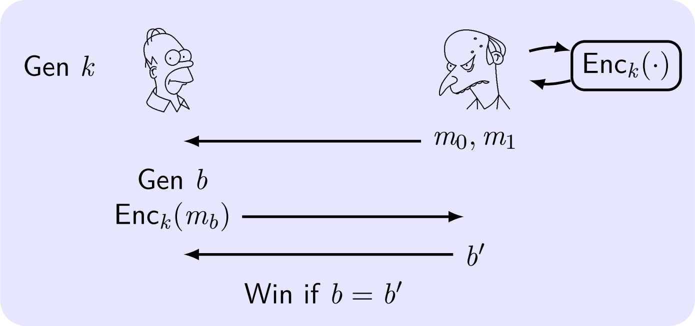
## A story in WWII

- Navy cryptanalysts believe the ciphertext "AF" means "Midway island" in Japanese messages
- But the general did not believe that Midway island would be attacked
- Navy cryptanalysts sent a plaintext that the freshwater supplies at Midway island were low
- Japanese intercepted the plaintext and sent a ciphertext that "AF" was low in water
- The US forces dispatched three aircraft carriers and won

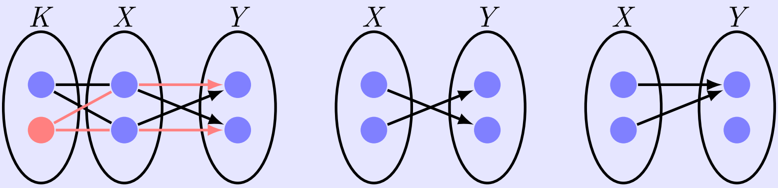
# Security Against CPA

The CPA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(n)$:

1. $k \leftarrow \mathsf{Gen}(1^n)$
2. $\mathcal{A}$ is given input $1^n$ and **oracle access** $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ to $\mathsf{Enc}_k(\cdot)$, outputs $m_0, m_1$ of the same length
3. $b \leftarrow \{0,1\}$. Then $c \leftarrow \mathsf{Enc}_k(m_b)$ is given to $\mathcal{A}$
4. $\mathcal{A}$ **continues to have oracle access** to $\mathsf{Enc}_k(\cdot)$, outputs $b'$
5. If $b' = b$, $\mathcal{A}$ succeeded $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi} = 1$, otherwise 0



# Concepts on Pseudorandom Functions




$F : K \times X \to Y$

2D function

$F_k(x) \stackrel{\text{def}}{=} F(k, x)$

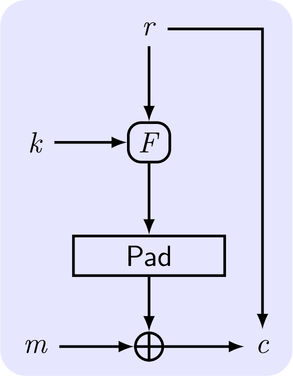
Keyed function

$f : X \to Y$

Look-up table

- **Keyed function** $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$
  $F_k : \{0,1\}^* \to \{0,1\}^*,\ F_k(x) \stackrel{\text{def}}{=} F(k, x)$
- **Look-up table** $f$: $\{0,1\}^n \to \{0,1\}^n$ with size $= ?$ bits
- **Function family** Func$_n$: all functions $\{0,1\}^n \to \{0,1\}^n$.
  $|\text{Func}_n| = 2^{n \cdot 2^n}$
- **Length Preserving**: $\ell_{key}(n) = \ell_{in}(n) = \ell_{out}(n)$

# CPA-Security from Pseudorandom Function




**Construction 7**

- *Fresh random string* $r$.
- $F_k(r)$: $|k| = |m| = |r| = n$.
- Gen: $k \in \{0,1\}^n$.
- Enc: $s := F_k(r) \oplus m$, $c := \langle r, s \rangle$.
- Dec: $m := F_k(r) \oplus s$.

**Theorem 8**

*If $F$ is a PRF, this fixed-length encryption scheme $\Pi$ is CPA-secure.*

# Pseudorandom Permutations

- **Bijection**: $F$ is one-to-one and onto
- **Permutation**: A bijective function from a set to itself
- **Keyed permutation**: $\forall k, F_k(\cdot)$ is permutation
- $F$ is a bijection $\iff$ $F^{-1}$ is a bijection

**Definition 9**

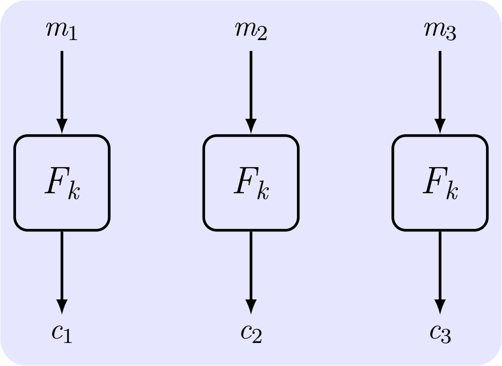
An efficient, keyed permutation $F$ is a **strong pseudorandom permutation (PRP)** if $\forall$ PPT distinguishers $D$,

$$\left|\Pr[D^{F_k(\cdot),F_k^{-1}(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot),f^{-1}(\cdot)}(1^n) = 1]\right| \leq \mathsf{negl}(n),$$

where $f$ is chosen *u.a.r* from the set of permutations on $n$-bit strings.

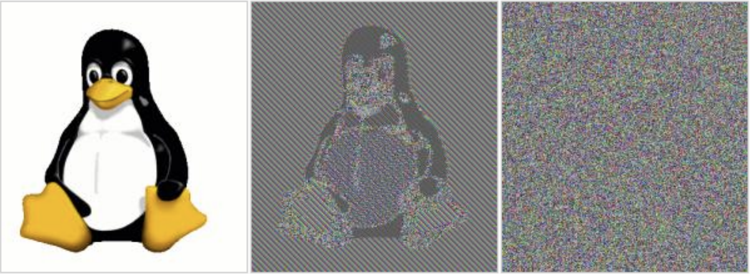
**If $F$ is a pseudorandom permutation then is it a PRF?**

# Electronic Code Book (ECB) Mode



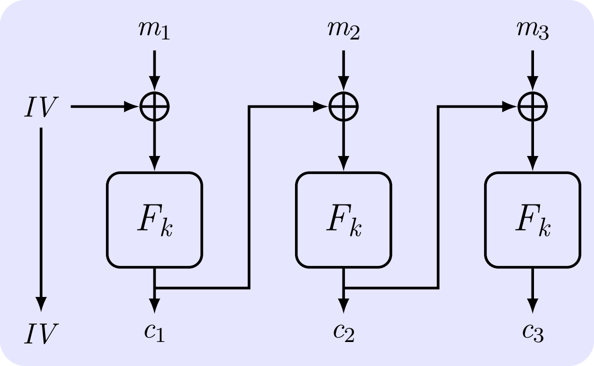

- Q: is it indistinguishable in the presence of an eavesdropper?
- Q: can $F$ be any PRF?

# Attack on ECB mode



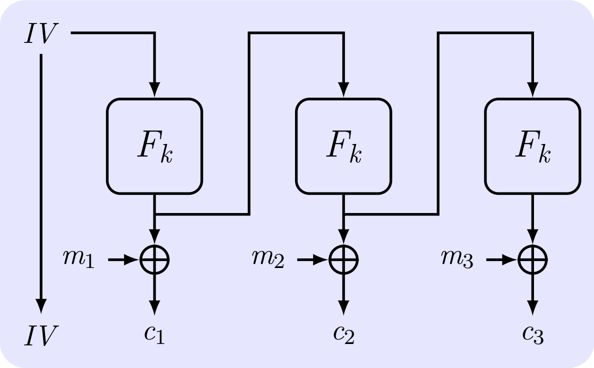
Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

# Cipher Block Chaining (CBC) Mode

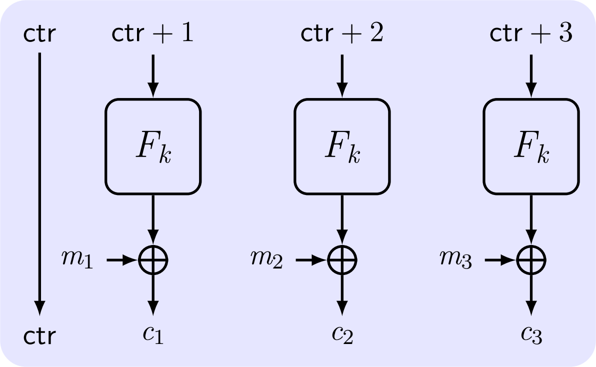



- $IV$: initial vector, a fresh random string.

# Output Feedback (OFB) Mode



# Counter (CTR) Mode



# $IV$ Should Not Be Predictable

If $IV$ is predictable, then CBC/OFB/CTR mode is not CPA-secure.

**Bug in SSL/TLS 1.0**

$IV$ for record $\#i$ is last CT block of record $\#(i-1)$.

**API in OpenSSL**

```
void AES_cbc_encrypt (
    const unsigned char *in,
    unsigned char       *out,
    size_t              length,
    const AES_KEY       *key,
    unsigned char       *ivec,    User supplies IV
    AES_ENCRYPT or AES_DECRYPT);
```

# Security Against CCA

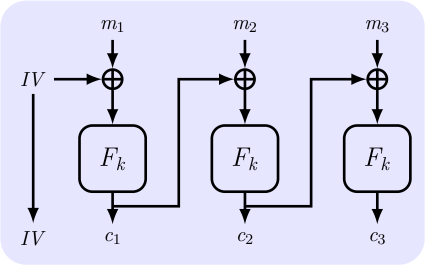
The CCA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$:

1. $k \leftarrow \mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ is given input $1^n$ and oracle access $\mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ and $\mathcal{A}^{\mathsf{Dec}_k(\cdot)}$, outputs $m_0, m_1$ of the same length.
3. $b \leftarrow \{0,1\}$. $c \leftarrow \mathsf{Enc}_k(m_b)$ is given to $\mathcal{A}$.
4. $\mathcal{A}$ continues to have oracle access **except for** $c$, outputs $b'$.
5. If $b' = b$, $\mathcal{A}$ succeeded $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi} = 1$, otherwise 0.

- In real world, the adversary might conduct CCA by influencing what gets decrypted
  - If the communication is not authenticated, then an adversary may send certain ciphertexts on behalf of the honest party
- CCA-security implies "**non-malleability**"
- None of the above scheme is CCA-secure

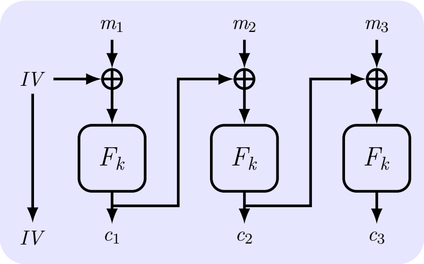
# Padding-Oracle Attacks

- In a one-block CBC, by modifying the 1st byte of $IV$, attacker can learn whether $m$ is NULL. If yes, error will occur.




- append $\{b\}^b$ as a dummy block if $m$ is NULL
- change the 1st byte of $IV$ from $x$ to $y$, get decrypted block $(x \oplus y \oplus b)\|\{b\}^{b-1}$, and trigger an error

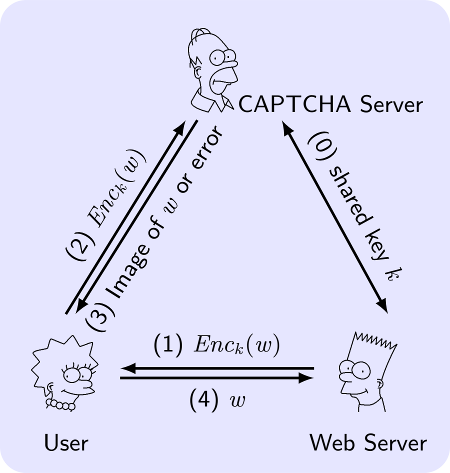
# Padding-Oracle Attacks (Cont.)




- If no error, then learn whether $m$ is 1 byte by modifying the 2nd byte of $IV$ and so on (changing the ciphertext)
- Once learn the length of $m$, learn the last byte of $m$ $(s)$ by modifying the one before the last block in the ciphertext
- $m_{last} = \cdots s \| \{b\}^b$, $c_{last-1} = \cdots t \| \{\cdot\}^b$
- modify $c_{last-1}$ to $c'_{last-1} = \cdots u \| (\{\cdot\}^b \oplus \{b\}^b \oplus \{b+1\}^b)$
- Q: If no padding error, then $s =$ ?

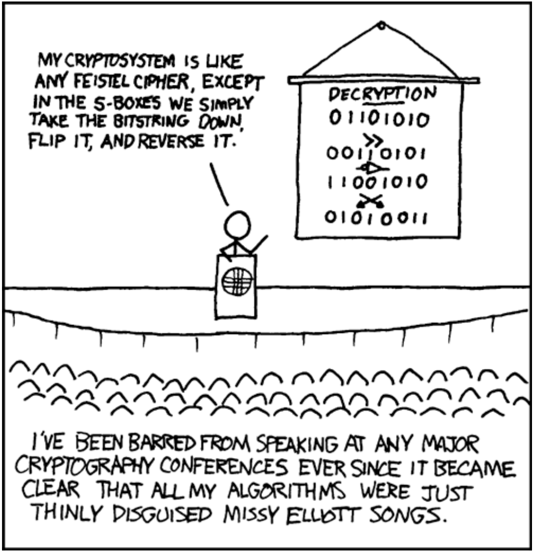
# Padding-Oracle Attacks: Real-world Case

CAPTCHA server will return an error when deciphering the CT of a CAPTCHA text received from a user.



# Comics on S-box [xkcd:153]

If you got a big keyspace, let me search it.



# Chronology of DES

**1973** NBS (NIST) publishes a call for a standard.

**1974** DES is published in the Federal Register.

**1977** DES is published as FIPS PUB 46.

**1990** Differential cryptanalysis with CPA of $2^{47}$ plaintexts.

**1997** DESCHALL Project breaks DES in public.

**1998** EFF's Deep Crack breaks DES in 56hr at $250,000.

**1999** Triple DES.

**2001** AES is published in FIPS PUB 197.

**2004** FIPS PUB 46-3 is withdrawn.

**2006** COPACOBANA breaks DES in 9 days at $10,000.

**2008** RIVYERA breaks DES within one day.

# AES – The Advanced Encryption Standard

- In 1997, NIST calls for AES.
- In 2001, Rijndael [J. Daemen & V. Rijmen] becomes AES.
- The first publicly accessible cipher for top secret information.
- Not only security, also efficiency and flexibility, etc.
- 128-bit block length and 128-, 192-, or 256-bit keys.
- Not a Feistel structure, but a SPN.
- Only non-trivial attacks are for reduced-round variants.
  - $2^{27}$ on 6-round of 10-round for 128-bit keys.
  - $2^{188}$ on 8-round of 12-round for 192-bit keys.
  - $2^{204}$ on 8-round of 14-round for 256-bit keys.

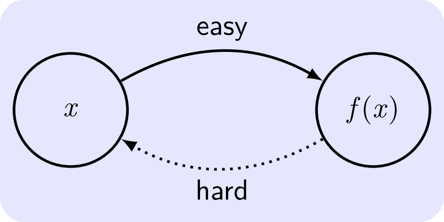
# Remarks on Block Ciphers

- **Block length** should be sufficiently large
- **Message tampering** is not with message confidentiality
- **Padding**: TLS: For $n > 0$, $n$ byte pad is $n, n, \ldots, n$ If no pad needed, add a dummy block
- **Stream ciphers vs. block ciphers**:
  - Steam ciphers are faster but have lower security
  - It is possible to use block ciphers in “stream-cipher mode”

**Performance: Crypto++ 5.6, AMD Opetron 2.2GHz**

| | Block/key size | Speed MB/sec |
|---|---|---|
| **RC4** | | 126 |
| **Salsa20/12** | | 643 |
| **Sosemanuk** | | 727 |
| **3DES** | 64/168 | 13 |
| **AES-128** | 128/128 | 109 |

# One-Way Functions (OWF)



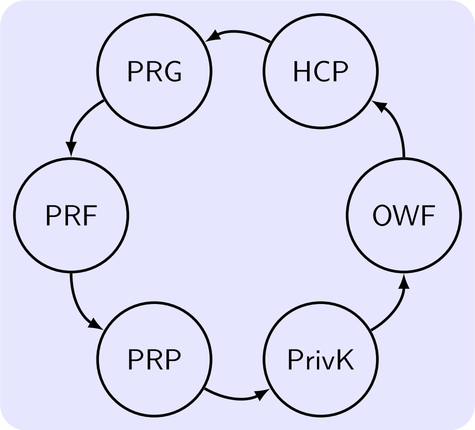

The inverting experiment $\mathsf{Invert}_{\mathcal{A},f}(n)$:

1. Choose input $x \leftarrow \{0,1\}^n$. Compute $y := f(x)$.
2. $\mathcal{A}$ is given $1^n$ and $y$ as input, and outputs $x'$.
3. $\mathsf{Invert}_{\mathcal{A},f}(n) = 1$ if $f(x') = y$, otherwise 0.

# Candidate One-Way Function

- **Multiplication and factoring**:
  $f_{\mathsf{mult}}(x, y) = (xy, \|x\|, \|y\|)$, $x$ and $y$ are equal-length primes.
- **Modular squaring and square roots**:
  $f_{\mathsf{square}}(x) = x^2 \bmod N$.
- **Discrete exponential and logarithm**:
  $f_{g,p}(x) = g^x \bmod p$.
- **Subset sum problem**:
  $f(x_1, \ldots, x_n, J) = (x_1, \ldots, x_n, \sum_{j \in J} x_j)$.
- **Cryptographically secure hash functions**:
  Practical solutions for one-way computation.

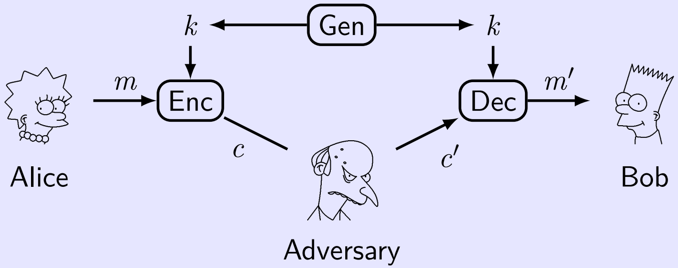
# Theoretical Constructions of Pseudorandom Objects

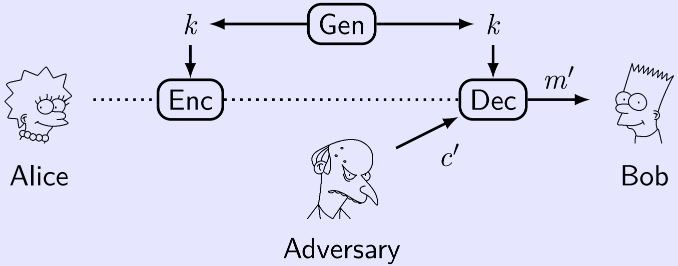



**One of contributions of modern cryptography**

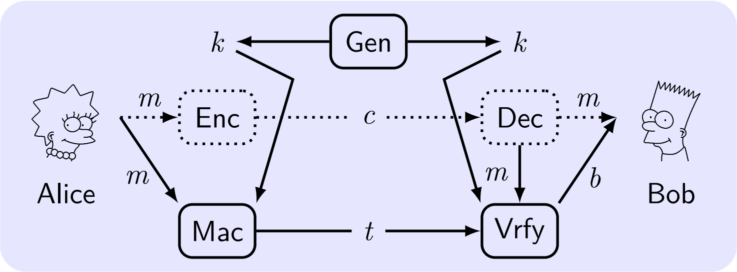
The existence of one-way functions is equivalent to the existence of all (non-trivial) private-key cryptography.

# Integrity and Authentication



# The Syntax of MAC




- key $k$, tag $t$, a bit $b$ means valid if $b = 1$; invalid if $b = 0$.
- **Key-generation** algorithm $k \leftarrow \mathsf{Gen}(1^n), |k| \geq n$.
- **Tag-generation** algorithm $t \leftarrow \mathsf{Mac}_k(m)$.
- **Verification** algorithm $b := \mathsf{Vrfy}_k(m, t)$.
- **Message authentication code**: $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$.
- **Basic correctness requirement**: $\mathsf{Vrfy}_k(m, \mathsf{Mac}_k(m)) = 1$.

# Security of MAC

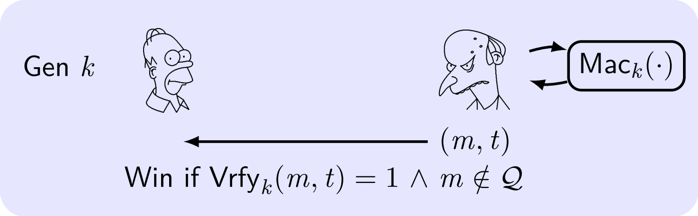
- **Intuition**: No adversary should be able to generate a **valid** tag on any "**new**" message[1] that was not previously sent.
- **Replay attack**: Copy a message and tag previously sent. (**excluded by only considering "new" message**)
  - Sequence numbers: receiver must store the previous ones.
  - Time-Stamps: sender/receiver maintain synchronized clocks.
- **Existential unforgeability**: **Not** be able to forge a valid tag on **any** message.
  - **Existential forgery**: *at least one* message.
  - **Selective forgery**: message chosen *prior* to the attack.
  - **Universal forgery**: *any* given message.
- **Adaptive chosen-message attack (CMA)**: be able to obtain tags on *any* message chosen adaptively *during* its attack.

[1] A stronger requirement is concerning *new message/tag pair*.

# Definition of MAC Security

The message authentication experiment $\mathsf{Macforge}_{\mathcal{A},\Pi}(n)$:

1. $k \leftarrow \mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ is given input $1^n$ and oracle access to $\mathsf{Mac}_k(\cdot)$, and outputs $(m, t)$. $\mathcal{Q}$ is the set of queries to its oracle.
3. $\mathsf{Macforge}_{\mathcal{A},\Pi}(n) = 1 \iff \mathsf{Vrfy}_k(m, t) = 1 \wedge m \notin \mathcal{Q}$.

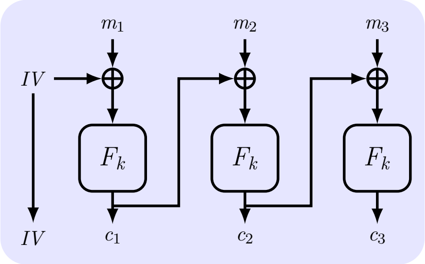


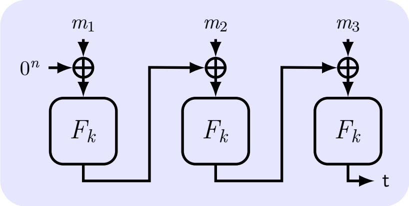

### Definition 10

A MAC $\Pi$ is **existentially unforgeable under an adaptive CMA** if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that:

$$\Pr[\mathsf{Macforge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

# Constructing Fixed-Length CBC-MAC



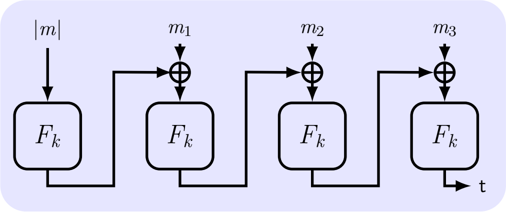




Modify CBC encryption into CBC-MAC:

- Change random $IV$ to encrypted fixed $0^n$,*otherwise*:
  Q: query $m_1$ and get $(IV, t_1)$; output $m_1' = IV' \oplus IV \oplus m_1$ and $t' =$ ____.
- Tag only includes the output of the final block,*otherwise*:
  Q: query $m_i$ and get $t_i$; output $m_i' = t_{i-1}' \oplus t_{i-1} \oplus m_i$ and $t_i' =$ ____.

# Secure Variable-Length MAC

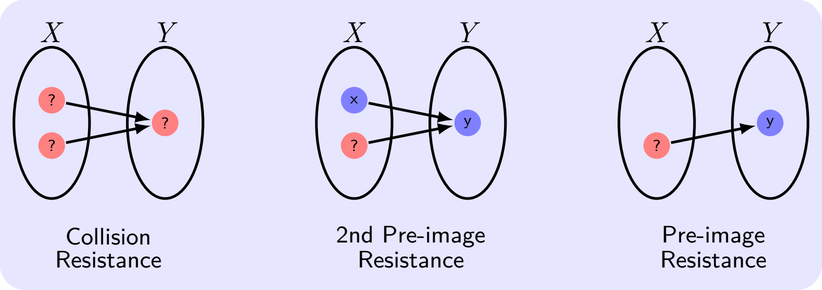
- **Input-length key separation**: $k_\ell := F_k(\ell)$, use $k_\ell$ for CBC-MAC.
- **Length-prepending**: Prepend $m$ with $|m|$, then use CBC-MAC.




- **Encrypt last block (ECBC-MAC)**: Use two keys $k_1, k_2$. Get $t$ with $k_1$ by CBC-MAC, then output $\hat{t} := F_{k_2}(t)$.

Q: To authenticate a voice stream, which approach do you prefer?

# Weaker Notions of Security for Hash Functions

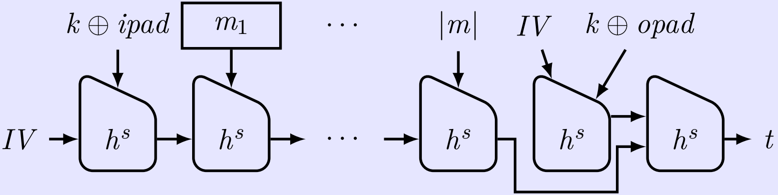



- **Collision resistance**: It is hard to find $(x, x'), x' \neq x$ such that $H(x) = H(x')$.
- **Second pre-image resistance**: Given $s$ and $x$, it is hard to find $x' \neq x$ such that $H^s(x') = H^s(x)$.
- **Pre-image resistance**: Given $s$ and $y = H^s(x)$, it is hard to find $x'$ such that $H^s(x') = y$.

# Applications of Hash Functions

- **Fingerprinting and Deduplication**: $H(alargefile)$ for virus fingerprinting, deduplication, P2P file sharing
- **Merkle Trees**: $H(H(H(file1), H(file2)), H(H(file3), H(file4)))$ fingerprinting multiple files / parts of a file
- **Passward Hashing**: $(salt, H(salt, pw))$ mitigating the risk of leaking password stored in the clear
- **Key Derivation**: $H(secret)$ deriving a key from a high-entropy (but not necessarily uniform) shared secret
- **Commitment Schemes**: $H(info)$ hiding the commited info; binding the commitment to a info

# Hash-based MAC (HMAC)




**Construction 11**

*$(\widetilde{\mathsf{Gen}}, h)$ is a fixed-length CRHF. $(\widetilde{\mathsf{Gen}}, H)$ is the Merkle-Damgård transform. $IV$, opad (0x36), ipad (0x5C) are fixed constants of length $n$. HMAC:*

- $\mathsf{Gen}(1^n)$: *Output* $(s, k)$. $s \leftarrow \widetilde{\mathsf{Gen}}, k \leftarrow \{0,1\}^n$ *u.a.r*
- $\mathsf{Mac}_{s,k}(m)$: $t := H^s_{IV}\Big((k \oplus \mathsf{opad}) \| H^s_{IV}\big((k \oplus \mathsf{ipad}) \| m\big)\Big)$
- $\mathsf{Vrfy}_{s,k}(m, t)$: $1 \iff t \stackrel{?}{=} \mathsf{Mac}_{s,k}(m)$

# Security of HMAC

**Theorem 12**

$$G(k) \stackrel{\text{def}}{=} h^s(IV \| (k \oplus \mathsf{opad})) \| h^s(IV \| (k \oplus \mathsf{ipad})) = k_1 \| k_2$$

$(\widetilde{\mathsf{Gen}}, h)$ *is CRHF. If* $G$ *is a PRG, then HMAC is secure.*

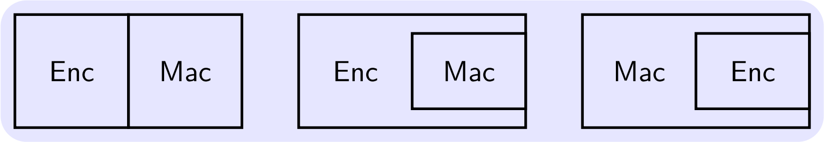
- HMAC is an industry standard (RFC2104)
- HMAC is faster than CBC-MAC
- Before HMAC, a common mistake was to use $H^s(k \| x)$
- Verification timing attacks: (Keyczar crypto library (Python))

  def Verify(key, msg, sig_bytes):

      return HMAC(key, msg) $==$ sig_bytes

  The problem: implemented as a byte-by-byte comparison
- *Don't implement it yourself*

# Combining Encryption and Authentication




- **Encrypt-and-authenticate** (e.g., SSH):

$$c \leftarrow \mathsf{Enc}_{k_1}(m),\ t \leftarrow \mathsf{Mac}_{k_2}(m).$$

- **Authenticate-then-encrypt** (e.g, SSL):

$$t \leftarrow \mathsf{Mac}_{k_2}(m),\ c \leftarrow \mathsf{Enc}_{k_1}(m \| t).$$

- **Encrypt-then-authenticate** (e.g, IPsec):

$$c \leftarrow \mathsf{Enc}_{k_1}(m),\ t \leftarrow \mathsf{Mac}_{k_2}(c).$$

# Analyzing Security of Combinations

**All-or-nothing**: Reject any combination for which there exists even a single counterexample is insecure.

- **Encrypt-and-authenticate**: $\mathsf{Mac}'_k(m) = (m, \mathsf{Mac}_k(m))$.
- **Authenticate-then-encrypt**:
  - $\text{Trans} : 0 \to 00; 1 \to 10/01$; $\mathsf{Enc}'$ uses CTR mode; $c = \mathsf{Enc}'(\mathsf{Trans}(m \| \mathsf{Mac}(m)))$.
  - Flip the first two bits of $c$ and verify whether the ciphertext is valid. $10/01 \to 01/10 \to 1$, $00 \to 11 \to \perp$.
  - If valid, the first bit of message is 1; otherwise 0.
  - For any MAC, this is not CCA-secure.
- **Encrypt-then-authenticate**:
  Decryption: If $\mathsf{Vrfy}(\cdot) = 1$, then $\mathsf{Dec}(\cdot)$; otherwise output $\perp$.

# Authenticated Encryption Theory and Practice

### Theorem 13

*$\Pi_E$ is CPA-secure and $\Pi_E$ is a secure MAC with unique tages, $\Pi'$ deriving from encrypt-then-authenticate approach is secure.*

**GCM(Galois/Counter Mode)**: CTR encryption then Galois MAC. (RFC4106/4543/5647/5288 on IPsec/SSH/TLS)
**EAX**: CTR encryption then CMAC.

### Proposition 14

*Authenticate-then-encrypt approach is secure if $\Pi_E$ is rand-CTR mode or rand-CBC mode.*

**CCM (Counter with CBC-MAC)**: CBC-MAC then CTR encryption. (802.11i, RFC3610)
**OCB (Offset Codebook Mode)**: integrating MAC into ENC. (two times fast as CCM, EAX)
**All support AEAD (A.E. with associated data):** part of message is in clear, and all is authenticated

# Remarks on Secure Message Transmission

- Authentication may leak the message.
- Secure message transmission implies CCA-security. The opposite direction is not necessarily true.
- Different security goals should always use different keys.
  - otherwise, the message may be leaked if $\mathsf{Mac}_k(c) = \mathsf{Dec}_k(c)$.
- Implementation may destroy the security proved by theory.
  - **Attack with padding oracle** (in TLS 1.0):
    **Dec** return two types of error: padding error, MAC error.
    **Adv.** learns last bytes if no padding error with guessed bytes.
  - **Attack non-atomic dec.** (in SSH Binary Packet Protocol):
    **Dec** (1)decrypt length field; (2)read packets as specified by the length; (3)check MAC.
    **Adv.** (1)send $c$; (2)send $l$ packets until "MAC error" occurs; (3)learn $l = \mathsf{Dec}(c)$.

# Password-Based KDF (PBKDF)

**Key stretching** increases the time of testing key (with slow hash function).

**Key strengthening** increases the length/randomness of key (with salt).

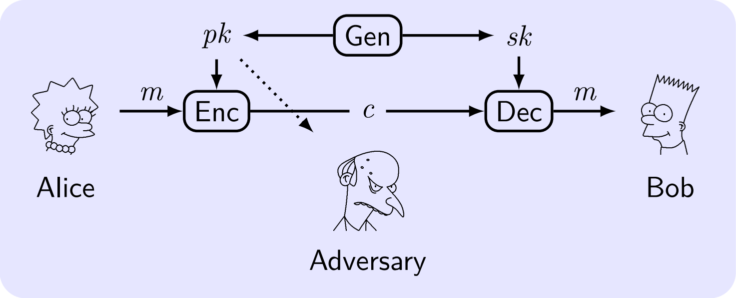
**PKCS#5 (PBKDF1)**: $H^{(c)}(pwd \| salt)$, iterate hash function $c$ times.

**Attack**: either try the enhanced key (larger key space), or else try the initial key (longer time per key).

# Public-Key Revolution

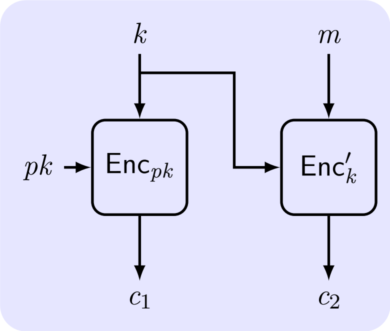
- In 1976, Whitfield Diffie and Martin Hellman published "*New Directions in Cryptography*".
- **Asymmetric** or **public-key** encryption schemes:
  - **Public key** as the encryption key.
  - **Private key** as the decryption key.
- **Public-key primitives**:
  - Public-key encryption.
  - Digital signatures. (non-repudiation)
  - Interactive key exchange.
- **Strength**:
  - Key distribution over public channels.
  - Reduce the need to store many keys.
  - Enable security in open system.
- **Weakness**: slow, active attack on public key distribution.

# Definitions




- **Key-generation** algorithm: $(pk, sk) \leftarrow$ Gen, key length $\geq n$.
- **Plaintext space** $\mathcal{M}$ is associated with $pk$.
- **Encryption** algorithm: $c \leftarrow \mathsf{Enc}_{pk}(m)$.
- **Decryption** algorithm: $m := \mathsf{Dec}_{sk}(c)$, or outputs $\perp$.
- **Requirement**: $\Pr[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m] \geq 1 - \mathsf{negl}(n)$.

To speed up the encryption of long message, use private-key encryption $\Pi'$ in tandem with public-key encryption $\Pi$.




**Construction 15**

$\Pi^{\mathsf{hy}} = (\mathsf{Gen}^{\mathsf{hy}}, \mathsf{Enc}^{\mathsf{hy}}, \mathsf{Dec}^{\mathsf{hy}})$:

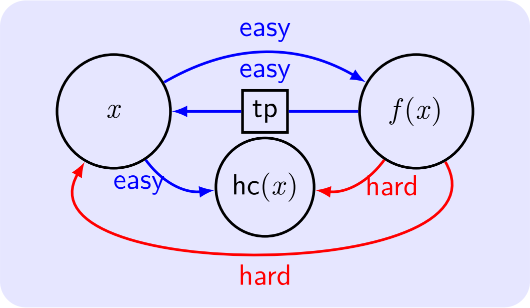
- $\mathsf{Gen}^{\mathsf{hy}}$: $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
- $\mathsf{Enc}^{\mathsf{hy}}$: $pk$ *and* $m$.
  1. $k \leftarrow \{0,1\}^n$.
  2. $c_1 \leftarrow \mathsf{Enc}_{pk}(k)$, $c_2 \leftarrow \mathsf{Enc}'_k(m)$.
- $\mathsf{Dec}^{\mathsf{hy}}$: $sk$ *and* $\langle c_1, c_2 \rangle$.
  1. $k := \mathsf{Dec}_{sk}(c_1)$.
  2. $m := \mathsf{Dec}'_k(c_2)$.

Q: is hybrid encryption a public-key enc. or private-key enc. ?

# Trapdoor Permutations

**Trapdoor function**: is easy to compute, yet difficult to find its inverse without special info., the "trapdoor". (One Way Function with the "trapdoor")

A public-key encryption scheme can be constructed from any trapdoor permutation. ("*Theory and Applications of Trapdoor Functions*", [Yao, 1982])



# Public-key Encryption Schemes from TDPs

**Construction 16**

- Gen: $(I, \mathsf{td}) \leftarrow \widehat{Gen}$ *output* ***public key*** $I$ *and* ***private key*** $\mathsf{td}$.
- Enc: *on input* $I$ *and* $m \in \{0,1\}$, *choose a random* $x \leftarrow \mathcal{D}_I$ *and output* $\langle f_I(x), \mathsf{hc}_I(x) \oplus m \rangle$.
- Dec: *on input* $\mathsf{td}$ *and* $\langle y, m' \rangle$, *compute* $x := f_I^{-1}(y)$ *and output* $\mathsf{hc}_I(x) \oplus m'$.

**Theorem 17**

*If* $\widehat{\Pi} = (\widehat{Gen}, f)$ *is TDP, and* $\mathsf{hc}$ *is HCP for* $\widehat{\Pi}$, *then Construction* $\Pi$ *is CPA-secure.*

**Is the following scheme is secure?**

$\mathsf{Enc}_I(m) = f_I(m)$, $\mathsf{Dec}_{\mathsf{td}}(c) = f_I^{-1}(c)$.

# Scenarios of CCA in Public-Key Setting

1. An adversary $\mathcal{A}$ observes the ciphertext $c$ sent by $\mathcal{S}$ to $\mathcal{R}$.
2. $\mathcal{A}$ send $c'$ to $\mathcal{R}$ in the name of $\mathcal{S}$ or its own.
3. $\mathcal{A}$ infer $m$ from the decryption of $c'$ to $m'$.

**Scenarios**

- **login to on-line bank with the password**: trial-and-error, learn info from the feedback of bank.
- **reply an e-mail with the quotation of decrypted text**.
- **malleability of ciphertexts**: e.g. doubling others' bids at an auction.

# State of the Art on CCA2-secure Encryption

- **Zero-Knowledge Proof**: complex, and impractical. (e.g., Dolev-Dwork-Naor)
- **Random Oracle** model: efficient, but not realistic (to consider CRHF as RO). (e.g., RSA-OAEP and Fujisaki-Okamoto)
- **DDH(Decisional Diffie-Hellman assumption) and UOWHF(Universal One-Way Hashs Function)**: x2 expansion in size, but security proved w/o RO or ZKP (e.g., Cramer-Shoup system).

**CCA2-secure implies Plaintext-aware**: an adversary cannot produce a valid ciphertext without "knowing" the plaintext.

**Open problem**

Constructing a CCA2-secure scheme based on RSA problem as efficient as "Textbook RSA".

# Private Key Encryption vs. Public Key Encryption

| | **Private Key** | **Public Key** |
|---|---|---|
| **Secret Key** | both parties | receiver |
| **Weakest Attack** | Eav | CPA |
| **Probabilistic** | CPA/CCA | always |
| **Assumption against CPA** | OWF | TDP |
| **Assumption against CCA** | OWF | TDP+RO |
| **Efficiency** | fast | slow |

# RSA Overview

- **RSA**: Ron Rivest, Adi Shamir and Leonard Adleman, in 1977
- **RSA problem**: Given $N = pq$ (two distinct big prime numbers) and $y \in \mathbb{Z}_N^*$, compute $y^{-e}$, $e^{\text{th}}$-root of $y$ modulo $N$
- Open problem:RSA problem is easier than factoring $N$?
- **Certification**: PKCS#1 (RFC3447), ANSI X9.31, IEEE 1363
- **Key sizes**: 1,024 to 4,096 bit
- **Best public cryptanalysis**: a 768 bit key has been broken
- **RSA Challenge**: break RSA-2048 to win $200,000 USD

**Key lengths** with comparable security :

| Symmetric | RSA |
|---|---|
| 80 bits | 1024 bits |
| 128 bits | 3072 bits |
| 256 bits | 15360 bits |

# “Textbook RSA”

### Construction 18

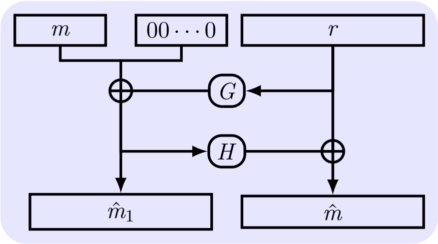
- Gen*: on input* $1^n$ *run* GenRSA$(1^n)$ *to obtain* $N, e, d$*.* $pk = \langle N, e \rangle$ *and* $sk = \langle N, d \rangle$*.*
- Enc*: on input* $pk$ *and* $m \in \mathbb{Z}_N^*$*,* $c := [m^e \bmod N]$*.*
- Dec*: on input* $sk$ *and* $m \in \mathbb{Z}_N^*$*,* $m := [c^d \bmod N]$*.*

### Insecurity

Since the “textbook RSA” is deterministic, it is insecure with respect to any of the definitions of security we have proposed.

# PKCK #1 v2.1 (RSAES-OAEP) (Cont.)

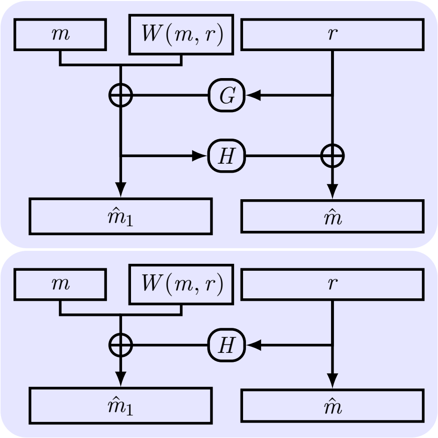
RSA-OAEP is CCA-secure in Random Oracle model. [2] [RFC 3447]




CPA: To learn $r$, attacker has to learn $\hat{m}_1$ from $(\hat{m}_1 \| \hat{m})^e$
CCA: Effective decryption query is disabled by checking "00...0" in the plaintext before the response

[2] It may not be secure when RO is instantiated.

# OAEP Improvements




**OAEP+**: $\forall$ trap-door permutation F, F-OAEP+ is CCA-secure.

**SAEP+**: RSA (e=3) is a trap-door permutation, RSA-SAEP+ is CCA-secure.

$W, G, H$ are Random Oracles.

### Simplified CCA on PKCS1 v1.5 in HTTPS [Bleichenbacher]

Server tells if the MSB of plaintext (Version Number) = '1' for a given ciphertext. Attacker sends $c' = (2^r)^e \cdot c$. If receiving $Yes$, then $(r+1)$-th $MSB(m) = ?$

$pk$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $sk$

$\xrightarrow{[c' = (2^r)^e \cdot c]}$

$\xleftarrow{[\text{Yes/No: } MSB(m') = \text{`1'}]}$

Client $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ Server

**Defense**: treating incorrectly formatted message blocks in a manner indistinguishable from correctly formatted blocks. See [RFC 5246]

# Implementation Attacks on RSA (Cont.)

**Timing attack**: [Kocher et al. 1997] The time it takes to compute $c^d$ can expose $d$. (require a high-resolution clock)
**Power attack**: [Kocher et al. 1999] The power consumption of a smartcard while it is computing $c^d$ can expose $d$.
**Defense**: **Blinding** by choosing a random $r$ and deciphering $r^e \cdot c$.

**Key generation trouble** (in OpenSSL RSA key generation):
Same $p$ will be generated by multiple devices (due to poor entropy at startup), but different $q$ (due to additional randomness).
Q: $N_1, N_2$ from different devices, $\gcd(N_1, N_2) = ?$
Experiment result: factor 0.4% of public HTTPS keys.

# Faults Attack on RSA

**Faults attack**: A computer error during $c^d \bmod N$ can expose $d$.

Using Chinese Remainder Theory to speed up the decryption:

$$[c^d \bmod N] \leftrightarrow ([m_p \equiv c^d \pmod p], [m_q \equiv c^d \pmod q]).$$

**Suppose error occurs when computing $m_q$, but no error in $m_p$.**

Then output $m' \equiv c^d \pmod p$, $m' \not\equiv c^d \pmod q$.
So $(m')^e \equiv c \pmod p$, $(m')^e \not\equiv c \pmod q$.

$$\gcd((m')^e - c, N) = ?$$

**Defense**: check output. (but 10% slowdown)

# Diffie-Hellman Assumptions

- **Computational Diffie-Hellman (CDH)** problem:

$$\mathsf{DH}_g(h_1, h_2) \stackrel{\mathsf{def}}{=} g^{\log_g h_1 \cdot \log_g h_2}$$

- **Decisional Diffie-Hellman (DDH)** problem:
  Distinguish $\mathsf{DH}_g(h_1, h_2)$ from a random group element $h'$.

**Definition 19**

DDH problem is hard relative to $\mathcal{G}$ if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that

$$|\Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]|$$

$$\leq \mathsf{negl}(n).$$

**Intractability of DL, CDH and DDH**

DDH is easier than CDH and DL.

# Diffie-Hellman Key-Exchange Protocol

$(\mathbb{G}, q, g) \leftarrow \mathcal{G}$

$x \leftarrow \mathbb{Z}_q$

$h_1 := g^x \xrightarrow{\mathbb{G}, q, g, h_1}$

$y \leftarrow \mathbb{Z}_q$

$\xleftarrow{h_2} h_2 := g^y$

$k_A := h_2^x$ $\qquad$ $k_B := h_1^y$

Q: $k_A = k_B = k = ?$

$\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}$ denote an experiment where if $b = 0$ the adversary is given $\hat{k} \leftarrow \mathbb{G}$.

**Theorem 20**

*If DDH problem is hard relative to $\mathcal{G}$, then DH key-exchange protocol $\Pi$ is secure in the presence of an eavesdropper (with respect to the modified experiment $\widehat{\mathsf{KE}}^{\mathsf{eav}}_{\mathcal{A},\Pi}$).*

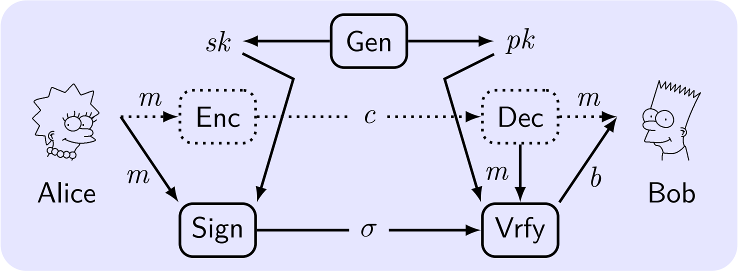
**Security**

Insecurity against active adversaries (Man-In-The-Middle).

# Digital Signatures – An Overview

- **Digital signature scheme** is a mathematical scheme for demonstrating the authenticity/integrity of a digital message
- allow a **signer** $S$ to "**sign**" a message with its own $sk$, anyone who knows $S$'s $pk$ can **verify** the authenticity/integrity
- (Comparing to MAC) digital signature is:
  - publicly verifiable
  - transferable
  - non-repudiation
  - but slow
- Q: What are the differences between digital signatures and handwritten signatures?
- Digital signature is NOT the "inverse" of public-key encryption

# The Syntax of Digital Signature Scheme




- signature $\sigma$, a bit $b$ means valid if $b = 1$; invalid if $b = 0$.
- **Key-generation** algorithm $(pk, sk) \leftarrow \mathsf{Gen}(1^n), |pk|, |sk| \geq n$.
- **Signing** algorithm $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$.
- **Verification** algorithm $b := \mathsf{Vrfy}_{pk}(m, \sigma)$.
- **Basic correctness requirement**: $\mathsf{Vrfy}_{pk}(m, \mathsf{Sign}_{sk}(m)) = 1$.

# Defining of Signature Security

The signature experiment $\mathsf{Sigforge}_{\mathcal{A},\Pi}(n)$:

1. $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ is given input $1^n$ and oracle access to $\mathsf{Sign}_{sk}(\cdot)$, and outputs $(m, \sigma)$. $\mathcal{Q}$ is the set of queries to its oracle.
3. $\mathsf{Sigforge}_{\mathcal{A},\Pi}(n) = 1 \iff \mathsf{Vrfy}_{pk}(m, \sigma) = 1 \wedge m \notin \mathcal{Q}$.

**Definition 21**

A signature scheme $\Pi$ is **existentially unforgeable under an adaptive CMA** if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that:

$$\Pr[\mathsf{Sigforge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

**Q: What's the difference on the ability of adversary between MAC and digital signature? What if an adversary is not limited to PPT?**

# The “Hash-and-Sign” Paradigm

## Construction 22

$\Pi = (\mathsf{Gen}_S, \mathsf{Sign}, \mathsf{Vrfy})$, $\Pi_H = (\mathsf{Gen}_H, H)$. *A signature scheme* $\Pi'$*:*

- $\mathsf{Gen}'$*: on input* $1^n$ *run* $\mathsf{Gen}_S(1^n)$ *to obtain* $(pk, sk)$*, and run* $\mathsf{Gen}_H(1^n)$ *to obtain* $s$*. The public key is* $pk' = \langle pk, s\rangle$ *and the private key is* $sk' = \langle sk, s\rangle$*.*
- $\mathsf{Sign}'$*: on input* $sk'$ *and* $m \in \{0,1\}^*$*,* $\sigma \leftarrow \mathsf{Sign}_{sk}(H^s(m))$*.*
- $\mathsf{Vrfy}'$*: on input* $pk'$*,* $m \in \{0,1\}^*$ *and* $\sigma$*, output 1* $\iff$ $\mathsf{Vrfy}_{pk}(H^s(m), \sigma) = 1$*.*

## Theorem 23

*If* $\Pi$ *is existentially unforgeable under an adaptive CMA and* $\Pi_H$ *is collision resistant, then Construction is existentially unforgeable under an adaptive CMA.*

# One-Time Signature (OTS)

**One-Time Signature (OTS)**: Under a weaker attack scenario, sign only one message with one secret.

The OTS experiment $\mathsf{Sigforge}^{\text{1-time}}_{\mathcal{A},\Pi}(n)$:

1. $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$.
2. $\mathcal{A}$ is given input $1^n$ and a single query $m'$ to $\mathsf{Sign}_{sk}(\cdot)$, and outputs $(m, \sigma)$, $m \neq m'$.
3. $\mathsf{Sigforge}^{\text{1-time}}_{\mathcal{A},\Pi}(n) = 1 \iff \mathsf{Vrfy}_{pk}(m, \sigma) = 1$.

**Definition 24**

A signature scheme $\Pi$ is **existentially unforgeable under a single-message attack** if $\forall$ PPT $\mathcal{A}$, $\exists$ negl such that:

$$\Pr[\mathsf{Sigforge}^{\text{1-time}}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

# Lamport's OTS

**Idea**: OTS from OWF; one mapping per bit.

### Construction 25

*$f$ is a one-way function.*

- Gen: *on input $1^n$, for $i \in \{1, \ldots, \ell\}$:*
  1. *choose random $x_{i,0}, x_{i,1} \leftarrow \{0,1\}^n$.*
  2. *compute $y_{i,0} := f(x_{i,0})$ and $y_{i,1} := f(x_{i,1})$.*

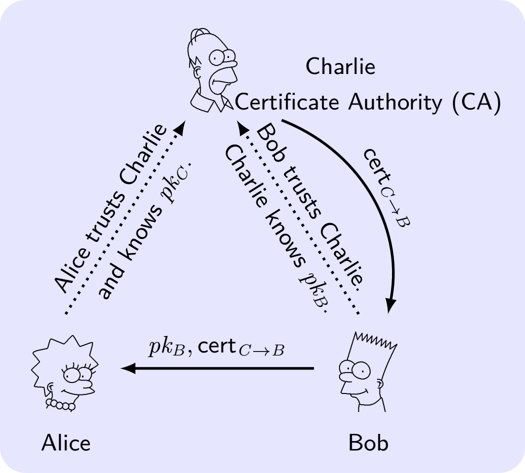
$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{\ell,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{\ell,1} \end{pmatrix} \quad sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{\ell,1} \end{pmatrix}.$$

- Sign*: $m = m_1 \cdots m_\ell$, output $\sigma = (x_{1,m_1}, \ldots, x_{\ell,m_\ell})$.*
- Vrfy*: $\sigma = (x_1, \ldots, x_\ell)$, output $1 \iff f(x_i) = y_{i,m_i}$, for all $i$.*

### Theorem 26

*If $f$ is OWF, $\Pi$ is OTS for messages of length polynomial $\ell$.*

# Certificates




**Certificates** $\text{cert}_{C \to B} \stackrel{\text{def}}{=} \text{Sign}_{sk_C}(\text{'Bob's key is } pk_B\text{'})$.

# Public-Key Infrastructure (PKI)

- **A single CA**: is trusted by everybody.
  - Strength: simple
  - Weakness: single-point-of-failure
- **Multiple CAs**: are trusted by everybody.
  - Strength: robust
  - Weakness: cannikin law
- **Delegation and certificate chains**: The trust is transitive.
  - Strength: ease the burden on the root CA.
  - Weakness: difficult for management, cannikin law.
- **"Web of trust"**: No central points of trust, e.g., PGP.
  - Strength: robust, work at "grass-roots" level.
  - Weakness: difficult to manage/give a guarantee on trust.

# Invalidating Certificates

- **Expiration**: include an *expiry date* in the certificate.

$$\text{cert}_{C\to B} \stackrel{\text{def}}{=} \text{Sign}_{sk_C}(\text{`bob's key is } pk_B\text{'}, \text{ date}).$$

- **Revocation**: explicitly revoke the certificate.

$$\text{cert}_{C\to B} \stackrel{\text{def}}{=} \text{Sign}_{sk_C}(\text{`bob's key is } pk_B\text{'}, \ \#\#\#).$$

"###" represents the serial number of this certificate.
**Cumulated Revocation**: CA generates *certificate revocation list* (CRL) containing the serial numbers of all revoked certificates, signs CRL with the current date.

# Simplified SSL/TLS Handshaking

**Purpose**: generate 4 secret keys with authenticated server
**Requirement**: the client has the public key of Trusted Third Party
the server has the certificate of its own $pk$ issued by TTP

 (Client)  (Server)

client random $a$ →

← server random $b$

← certificate of $pk$

$E_{pk}$(premaster secret $s$) →

Hash of previous msgs →

← Hash of previous msgs

session keys from $(a, b, s)$ | session keys from $(a, b, s)$

# Three-Pass Protocol

**Purpose**: communication without shared keys
**Requirement**: $\mathsf{Dec}_{k_1}(\mathsf{Enc}_{k_2}(\mathsf{Enc}_{k_1}(m))) = \mathsf{Enc}_{k_2}(m)$
**Shamir Protocol**: $p$ is a prime, find $e, d$ with $\gcd(e, p-1) = 1$ and $ed \equiv 1 \pmod{p-1}$

| | | |
|---|---|---|
| $p, e_A, d_A$ | | $p, e_B, d_B$ |
| $c = [m^{e_A} \bmod p]$ | $\longrightarrow$ | |
| | $\longleftarrow$ | $c_1 = [c^{e_B} \bmod p]$ |
| $c_2 = [c_1^{d_A} \bmod p]$ | $\longrightarrow$ | |
| | | $m = [c_2^{d_B} \bmod p]$ |

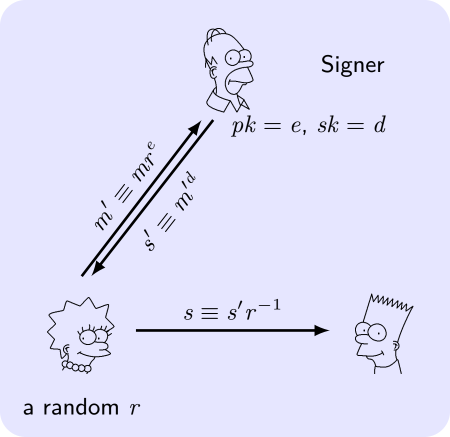
$$c_2^{d_B} = c_1^{d_A \cdot d_B} = c^{e_B \cdot d_A \cdot d_B} = m^{e_A \cdot e_B \cdot d_A \cdot d_B} = m^{e_A d_A \cdot e_B d_B} = m$$

**Weakness**: insecurity under the man-in-the-middle attack

# Blind Signature

**Blind signature** is a form of digital signature in which the message is blinded before it is signed.

Alice asks for Signer to sign $m$ blindly and then sends to Bob




$$s \equiv s' r^{-1} \equiv m'^d r^{-1} \equiv (mr^e)^d r^{-1} \equiv m^d.$$

# Group Signature

**Group Signature**: allowing a member of a group to anonymously sign a message on behalf of the group (with a group manager)

- **Soundness**: valid sigs by members verify correctly
- **Unforaeable**: only members can create valid sigs
- **Anonymity**: signer can be determined only by manager
- **Traceability**: manager can trace which member signed
- **Unlinkability**: cannot tell if two sigs were from same signer
- **Exculpability**: cannot forge a sig for other/non members

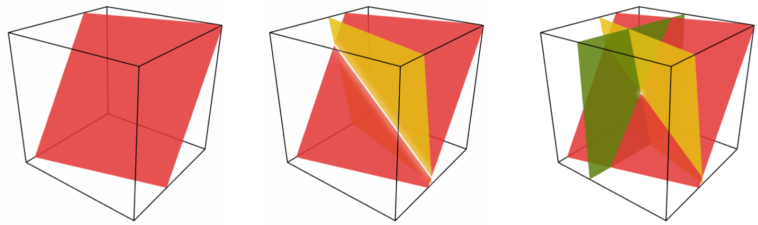
**A trivial group signature with trusted manager GM**:

- **KeyGen**: GM generates a secret key list for each member and publishes all of public keys
- **Sign**: sign with an unused secret key
- **Verify**: try all of public keys

# Secret Sharing

**Purpose**: distribute a secret amongst a group of $n$ participants, each of whom is allocated a share of the secret. The secret can be reconstructed only when a sufficient number of shares $t$ are combined together. It is called $(t, n)$-**threshold scheme**.

**Blakley's scheme**: any $n$ nonparallel $n$-dimensional hyperplanes intersect at a specific point.



**Chinese remainder theorem**: the shares of secret are generated by reduction modulo some relatively prime integers, and the secret is recovered by solving the system of congruences using the CRT.

# Threshold Cryptography

$(t, n)$-**threshold scheme**: at least $t$ of parties can efficiently decrypt/sign the ciphertext, while less than $t$ have no useful information

**Threshold Elgamal Cryptosystem**:

- **Key sharing**: $sk = s, pk = h = g^s$. Party $i$ obtains a share $s_i$ with Shamir's scheme ($(t, n)$-threshold secret sharing) such that $s = \Sigma_i s_i \cdot \lambda_i$ with public info $\lambda_i$ and publishes $h_i = g^{s_i}$
- **Enc**: $y \leftarrow \mathbb{Z}_q$, $\langle c_1, c_2 \rangle = \langle g^y, h^y \cdot m \rangle$
- **Dec**: Party $i$ outputs $d_i = c_1^{s_i}$ and ZKP of $\log_g h_i = \log_{c_1} d_i$

$$m = c_2 / \Pi_i d_i^{\lambda_i}$$

$$c_2 / \Pi_i d_i^{\lambda_i} = c_2 / \Pi_i c_1^{s_i \cdot \lambda_i} = c_2 / c_1^{\Sigma_i s_i \cdot \lambda_i} = c_2 / c_1^s = m$$

# Commitment Scheme

**Commitment scheme** allows one to commit to a value while keeping it hidden, with the ability to reveal the committed value

**Coin flipping over Internet**: Alice flips the coin, Bob guesses

$b \leftarrow \{H, T\}$

a random $r$

$m = b \| r$

$h = \text{Hash}(m) \longrightarrow$

$\longleftarrow b' \in \{H, T\}$

$m \longrightarrow$

$h \stackrel{?}{=} \text{Hash}(m)$

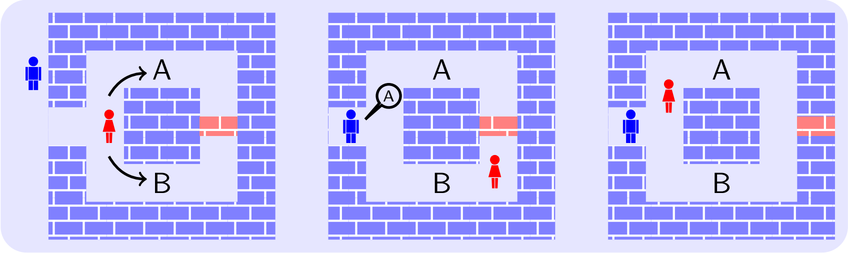
Win if $b = b'$

# Zero-Knowledge Proof

- **Interactive proof system** is an abstract machine that models computation as the exchange of messages between two parties: verifier and prover
- **Proof of knowledge**: an interactive proof in which **prover** succeeds convincing **verifier** that it knows something
- **Zero-knowledge proof (ZKP)**: an interactive proof *without revealing anything other than the veracity of the statement*
  - **Completeness**: if the statement is true, the honest "verifier" will be convinced by an honest prover
  - **Soundness**: if the statement is false, no cheating prover can convince the honest verifier
  - **Existence**: If OWF exists, ZKP exists for any NP-set

# A Toy Example of ZKP

Alice proves to Bob that she knows the secret word used to open a magic door in a cave



# Schnorr Protocol

**Schnorr protocol**: Alice proves to Bob the knowledge of $x = \log_g y$ in the discrete log problem

a random $r$

$t = g^r \longrightarrow$

$\longleftarrow$ a random bit $c \leftarrow \{0, 1\}$

$s = r + cx \longrightarrow$

$g^s \stackrel{?}{=} ty^c$

If Alice can foresee $c$, Alice can cheat with $t = g^s / y$ when $c = 1$.

# ZKP of the Ability to Break RSA

**Purpose**: Alice convinces Bob that she knows Charlie's private key $d$ for RSA problem $\langle N, e, d \rangle$, but she doesn't want to tell Bob $d$

$$\xleftrightarrow[ks \equiv e \pmod{N}]{\text{random } k, s > 3}$$

$$\xleftrightarrow{\text{random } c}$$

$m = c^d$

$x = m^k \longrightarrow$

$x^s \stackrel{?}{=} c$

If Alice can manipulate $c$, Alice can cheat with $c = m^e$.

**Oblivious transfer (OT)** protocol: a sender remains oblivious as to whether or which info has been transferred.
**Rabin's OT protocol**: RSA problem $\langle N, e, d\rangle$

$N, e, m^e \bmod N \longrightarrow$

a random $x$

$\longleftarrow x^2 \bmod N$

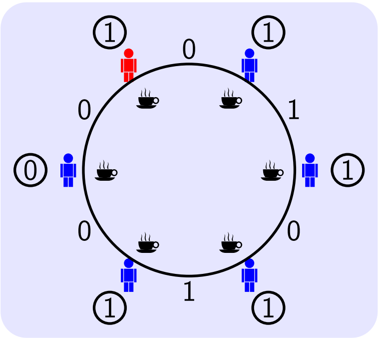
the square root $y$ of $x^2 \longrightarrow$

$y \stackrel{?}{=} \pm x$

If $y \neq \pm x$, then Bob can factorize $N$ with $\gcd(y - x, N)$ and find $d$. Since every quadratic residue modulo $N$ has four square roots, Bob can learn $m$ with probability $\frac{1}{2}$.

**Secure multi-party computation (MPC)**: enable parties to jointly compute a function over their inputs, while at the same time keeping these inputs private

**Dining Cryptographers Problem**: how to perform a secure MPC of the boolean-OR function




- at most one  (1), other  (0)
- every two adjacent people establish a shared one-bit secret
- everyone shouts the XOR of two shared secrets and its own bit
- output the XOR of all of what everyone shouts. If $1$, there is a , otherwise there is none

# Why Quantum Cryptography?

Quantum cryptography taps the natural uncertainty of the quantum world

- **Superposition**: object doesn't have definite properties (location, speed) but has probabilities over them
- **Interference**: probabilities can be negative
- **Entanglement**: properties of many particles can be correlated
- **Measurement**: object's properties collapse to definite value when measured, collapsing also properties of other entangled objects

# State-of-the-Art of Quantum Cryptography

- (Unsurprisingly) there is **no proof** that quantum computers are more powerful than classical computers/Boolean circuits/Turing machines
- There are **polynomial** algorithms for quantum computers solving problems unknown to be solvable classically in poly-time: factoring and discrete logs
- There are **hard** problem with no quantum poly-time algorithm: NPC, inverting many candidate OWF, private key encryption and signature schemes

# Quantum Key Distribution

**Purpose**: Using photon polarization states to transmit the information in a public channel against eavesdroppers

## BB84 protocol: C. H. Bennett and G. Brassard (1984)

| Basis | 0 | 1 |
|---|---|---|
| + | \| | – |
| x | / | \ |

| | |
|---|---|
| Alice's random bits | 01101001 |
| Alice's random sending basis | ++x+xxx+ |
| Photon polarization Alice sends | \|–\\|\//– |
| Bob's random measuring basis | +xxx+x++ |
| Photon polarization Bob measures | \|/\/–/–– |
| Shared secret key | 0 1  0 1 |

- Two bases are public
- Eavesdropping would change the photon polarization states
- Check for the presence of eavesdropping by comparing a subset of shared bit string

# Provable Security

- A proof of security never proves security in an absolute sense, it relates security to an unproven assumption that some computational problem is hard.
- The quality of a security reduction should not be ignored – it matters how tight it is, and how strong the underlying assumption is.
- A security reduction only proves something in a particular model specifying what the adversary has access to and can do.

# Crypto Pitfalls

Crypto deceptively simple

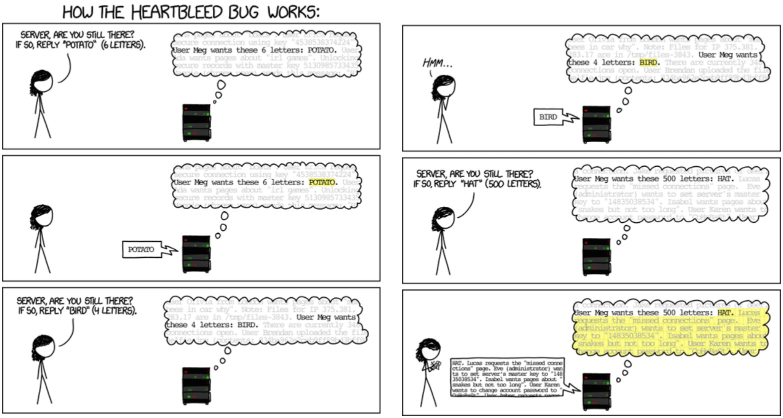
- Why does it so often fail?

Important to distinguish various issues:

1. Bad cryptography/implementations/design, etc.
2. Good cryptography can be ‘circumvented’ by adversaries operating ‘outside the model’
3. Even the best cryptography only shifts the weakest point of failure to elsewhere in your system
4. Systems are complex: key management; social engineering; insider attacks

Avoid the first; be aware of 2-4.

# Bad Implementation Example: Heartbleed



# Crypto is difficult to get right

- Must be implemented correctly
- Must be integrated from the beginning, not added on “after the fact”
- Need expertise; “a little knowledge can be a dangerous thing”
- Can’t be secured by Q/A, only (at best) through penetration testing and dedicated review of the code by security experts

# Beware of Snake Oil

**Snake Oil**: bogus commercial cryptographic products.

- **Secret system**: security through obscurity
- **Technobabble**: since cryptography is complicated
- **Unbreakable**: a sure sign of snake oil
- **One-time pads**: a flawed implementation
- **Unsubstantiated "bit" claims**: key lengths are not directly comparable

# General Recommendation

- Use only standardized algorithms and protocols
- No security through obscurity!
- Use primitives for their intended purpose
- Don't implement your own crypto
- If your system cannot use "off-the-shelf" crypto components, re-think your system
- If you really need something new, have it designed and/or evaluated by an expert
- Don't use the same key for multiple purposes
- Use good random-number generation

# Crypto Libraries

- Use existing, high-level crypto libraries: cryptlib, NaCl, Google's Keyczar, Mozilla's NSS, OpenSSL
- Avoid low-level libraries (like JCE, crypto++, GnuPG, OpenPGP) - too much possibility of mis-use
- Avoid writing your own low-level crypto