Introduction to Symmetric-Key Cryptography (Part-1)

ACM India Summer School 2024 (Day 2, Post-Lunch)

Sikhar Patranabis (IBM Research India)

sikhar.patranabis@ibm.com

https://research.ibm.com/people/sikhar-patranabis

IITB Trust Lab

4th June 2024

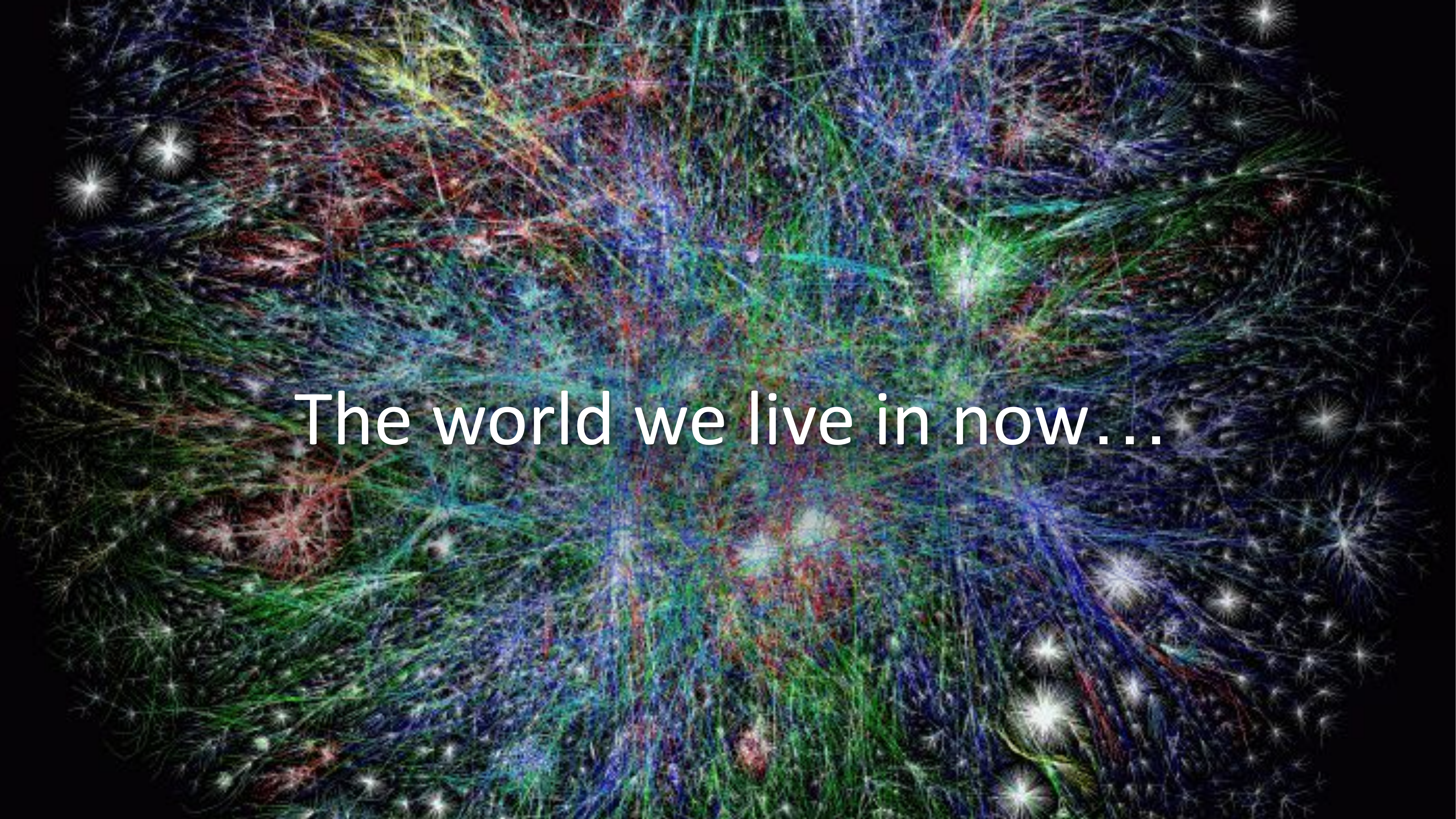
IBM Research

Scene setting

- **Cryptography: the art of secret writing.**
- Derived from the Greek:
  - kryptos (meaning ``hidden'')
  - grafo (meaning ``write“)
- **Used for centuries** by parties wishing to **communicate securely.**
  - **Historically**, associated with **encryption** (to provide confidentiality).
  - Now a significant area at the intersection between **CS, mathematics, and systems engineering**, and plays a crucial role in **information security**.
  - No longer limited to confidentiality, no longer limited to communications.


The world we used to live in…

The world changed…


The world we live in now…

Outline

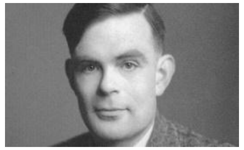
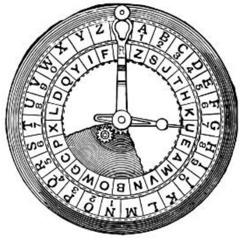
- Some practical perspectives on cryptography
- Secure communication
- Computational cryptography – one-way functions
- Pseudorandom generators (PRGs) and stream ciphers

Outline

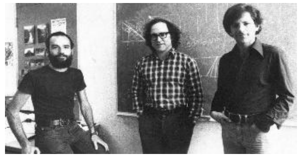
- Some practical perspectives on cryptography
- Secure communication
- Computational cryptography – one-way functions
- Pseudorandom generators (PRGs) and stream ciphers

Cryptography – past, present, future

Art of secure communication

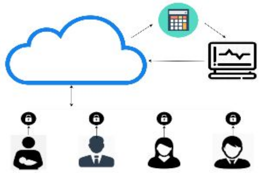

New directions and formalizations


Engineering and technology translation







Pre-1970 → 1970s and 1980s → Post-2000

Individual perspective

- **Cryptography makes our lives more convenient**.
  - Enables shopping online, communicating remotely with friends and family, interacting with government services online, working from home, etc.

- **Enables established human rights** such as privacy and freedom of expression.
  - In a free society, individuals should probably have the right to use cryptography in any way they see fit.
  - Indeed, a stated commitment to this freedom is one indicator of a truly free society.

Business perspective

- **Cryptography enables new forms of business**.
  - Allows provisioning of security services.
  - Ensures regulatory compliance for existing forms of business.

- **Cryptography may also bring new costs to a business**.
  - Moving business online may introduce new threats… which cryptography can only partially address, or which it does not address at all.

- **Cryptography will only be deployed if it makes business sense**.
  - Cost-effective, appropriate, compliant with regulations.

Government perspective

Conflicting requirements with respect to cryptography

- Cryptography as an enabler
    - Promotes a competitive and attractive business environment.
    - Enables streamlining operations by moving services on-line.
- Cryptography as a detractor
    - Control crime and manage issues of national security.
    - Limit the use of cryptography -- imposition of laws and regulations, promotion of weak cryptographic standards, or by other means.

Government perspective (example)

Our vision is for the UK in 2015 to derive huge economic and social value from a vibrant, resilient and secure cyberspace, where our actions, guided by our core values of liberty, fairness, transparency and the rule of law, enhance prosperity, national security and a strong society.

UK government cyber strategy, Nov 2011

https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/60961/uk-cyber-security-strategy-final.pdf

Our vision for 2021 is that the UK is secure and resilient to cyber threats, prosperous and confident in the digital world.

UK government cyber strategy, 2016

https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/567242/national_cyber_security_strategy_2016.pdf

The Indian context

THE PULSE | SECURITY | SOUTH ASIA

Securing India's Digital Future: Cybersecurity Urgency and Opportunities

Despite the looming specter of cyber attacks in India, there exists untapped potential and opportunities that the nation can harness to bolster cybersecurity.

certin
Enhancing Cyber Security in India

साइबर स्वच्छता केन्द्र
CYBER SWACHHTA KENDRA
Botnet Cleaning and Malware Analysis Centre

सत्यमेव जयते
Ministry of Electronics and Information Technology
Government of India

Home › News › Opinion

Digital India Act: Here's how it should fix India's cybersecurity weaknesses

Amid regular reports of government and large priv cyberattacks and data breaches, the DIA must be safeguarding privacy, data protection and cyberse frameworks between institutions, voluntary repor a large cadre of cybersecurity professionals

Why is Cybersecurity Important in Digital India?

India's rapid digitalization makes robust cybersecurity even more critical. Here's why:

- **Growing Reliance on Digital Infrastructure:** As online services become the norm, robust cybersecurity measures are essential to protect sensitive data like financial records and government information.
- **Increasing Internet Users:** With a rapidly growing internet user base, India presents a larger target for cybercriminals.
- **Evolving Threats:** Cyber threats are constantly evolving, so staying informed and adapting your defenses is crucial.

The importance of security infrastructure

- **Security infrastructure** *must* support deployment of cryptographic solutions.
  - **Infrastructure**: procedures, plans, policies, and management to ensure any deployment serves its intended purpose.

- **Cryptography on its own is not a magic bullet.**
  - Cryptography $\not\supseteq$ cryptographic algorithms.

- This is a principle worth keeping in mind throughout the course.

Modern approach to cryptography

Formal Security Model

Define an adversarial model

Application

Modern approach to cryptography

Cryptographic Solution

Formal Security Model

Define an adversarial model

Application

Cryptographic algorithms vs cryptographic implementations

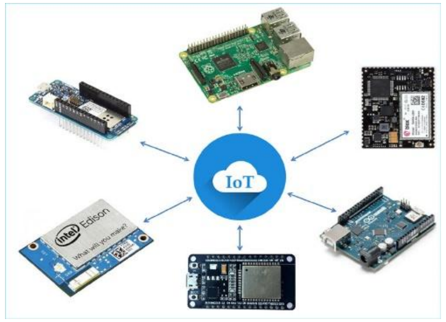
Cryptographic Solution

Formal Security Model

Define an adversarial model

Application

Deploy on Real Systems (Software/Hardware)



Cryptographic algorithms vs cryptographic implementations

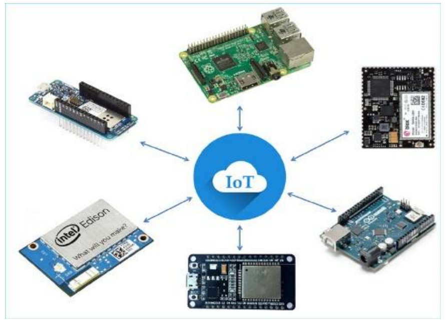
Implementation-Level Threats and Attacks

Formal Security Model

Define an adversarial model

Often **does not** capture implementation-level attacks

Deploy on Real Systems (Software/Hardware)



Cryptographic algorithms vs cryptographic implementations

Implementation-Level Threats and Attacks

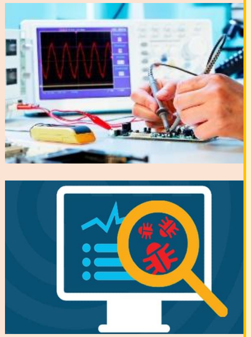
Formal Security Model

Define an adversarial model

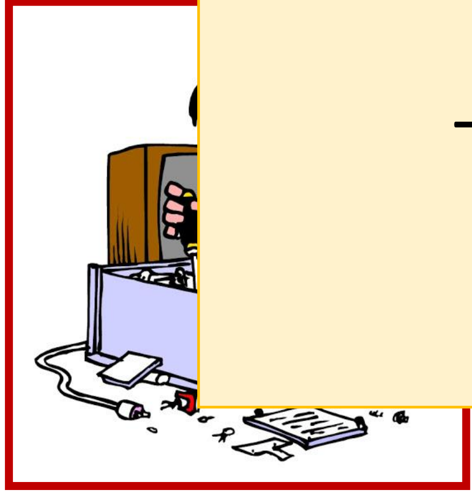
Target **insecure implementations** of provably secure cryptoprimitives

Cryptographic algorithms vs cryptographic implementations

Implementation-Level Threats and Attacks



Formal Security Model



“Cryptographers rarely sleep well”

- Silvio Micali (A.M. Turing Award Laureate, 2013)

Modern approach to cryptography

Cryptographic Solution

Formal Security Model

Define an adversarial model

Application

For this series of lectures, suffices to restrict to cryptographic algorithms

Modern approach to cryptography

Cryptographic Solution

Formal Security Model

Define an adversarial model

Application

For this series of lectures, suffices to restrict to cryptographic algorithms

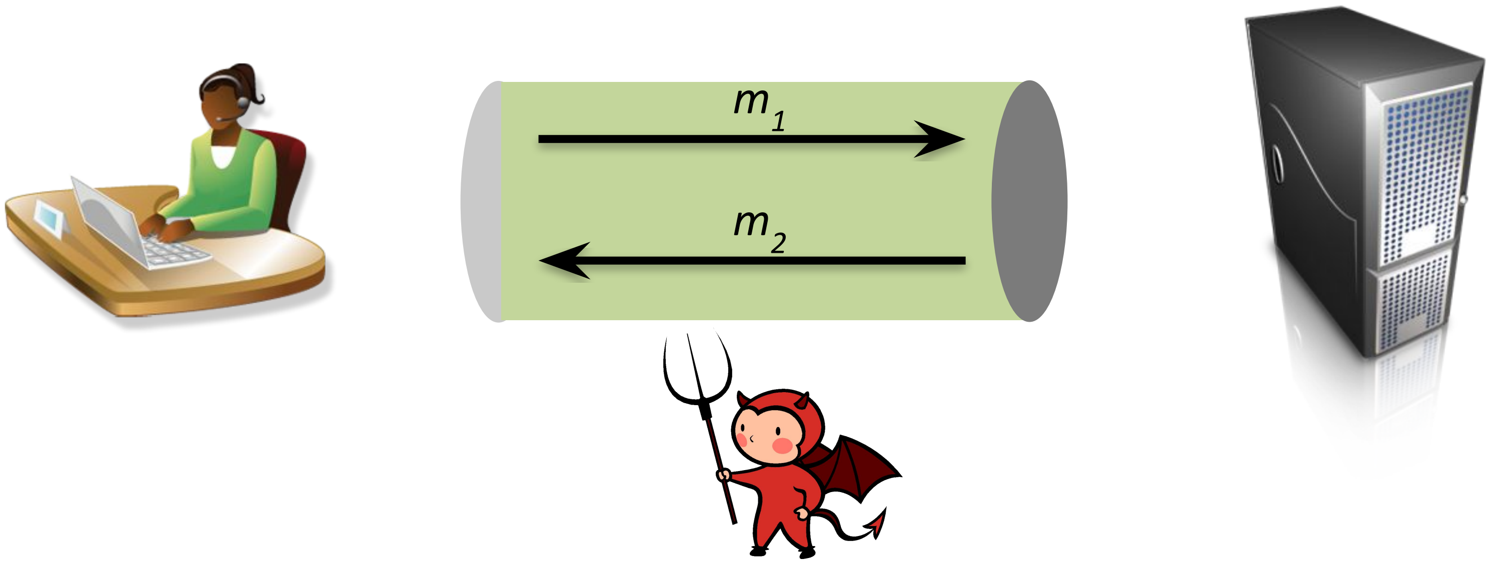
Application for next few lectures: **secure communication**

Outline

- Some practical perspectives on cryptography
- Secure communication
- Computational cryptography – one-way functions
- Pseudorandom generators (PRGs) and stream ciphers

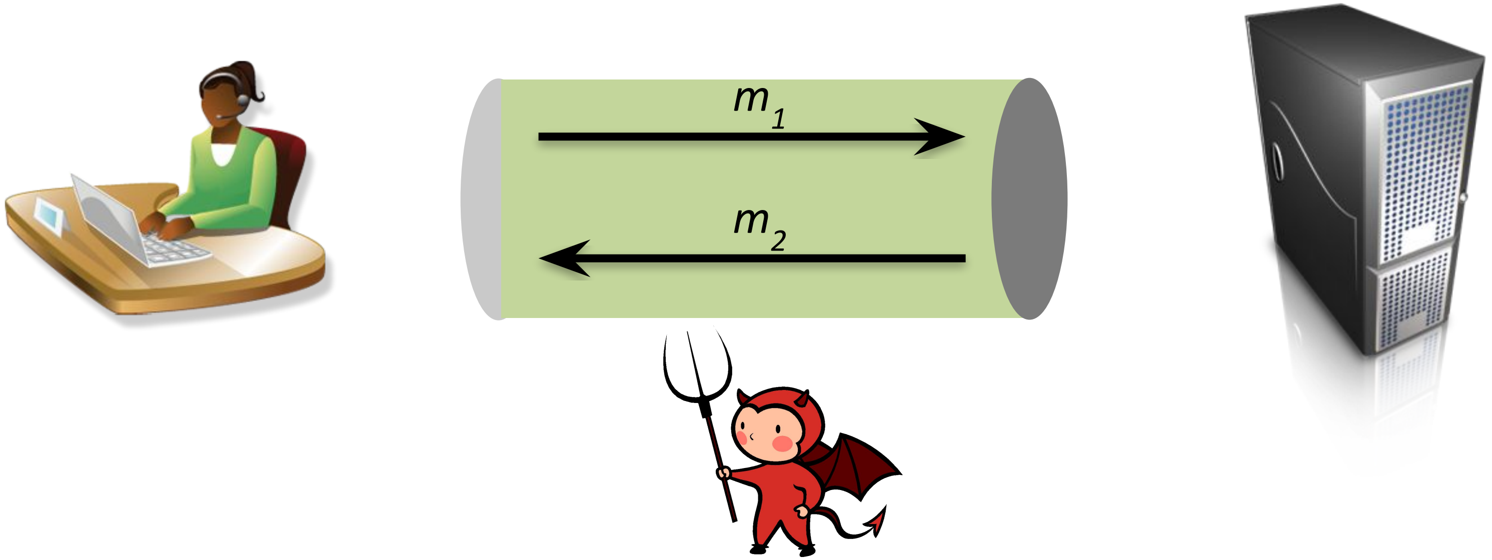
Crypto ground-zero: secure communication

Secure communication



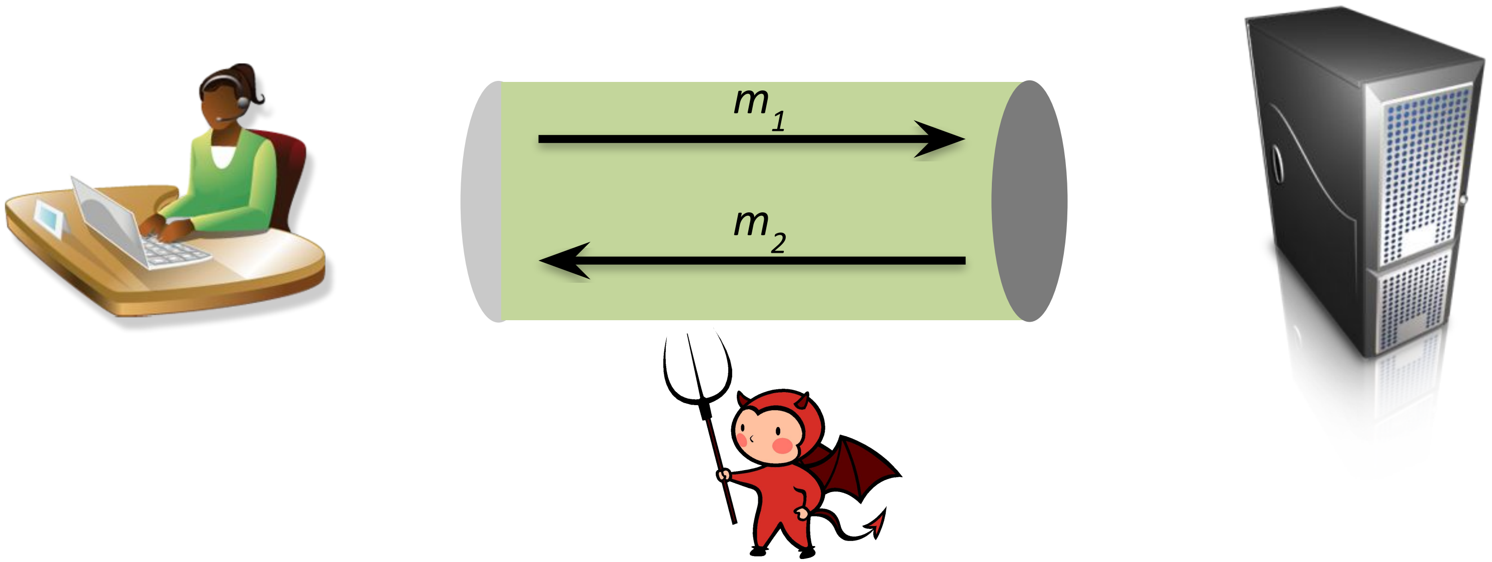
- Two entities exchange messages over an **insecure channel**.
  - Entities are often called **Alice and Bob** (but they need not be people).
- The insecure channel will be provided by a communications network.
  - Examples: wireless LAN, mobile phone network, “the Internet”, or a combination of these.
- **Use cryptography to build a secure channel on top of the insecure channel.**

Secure communication



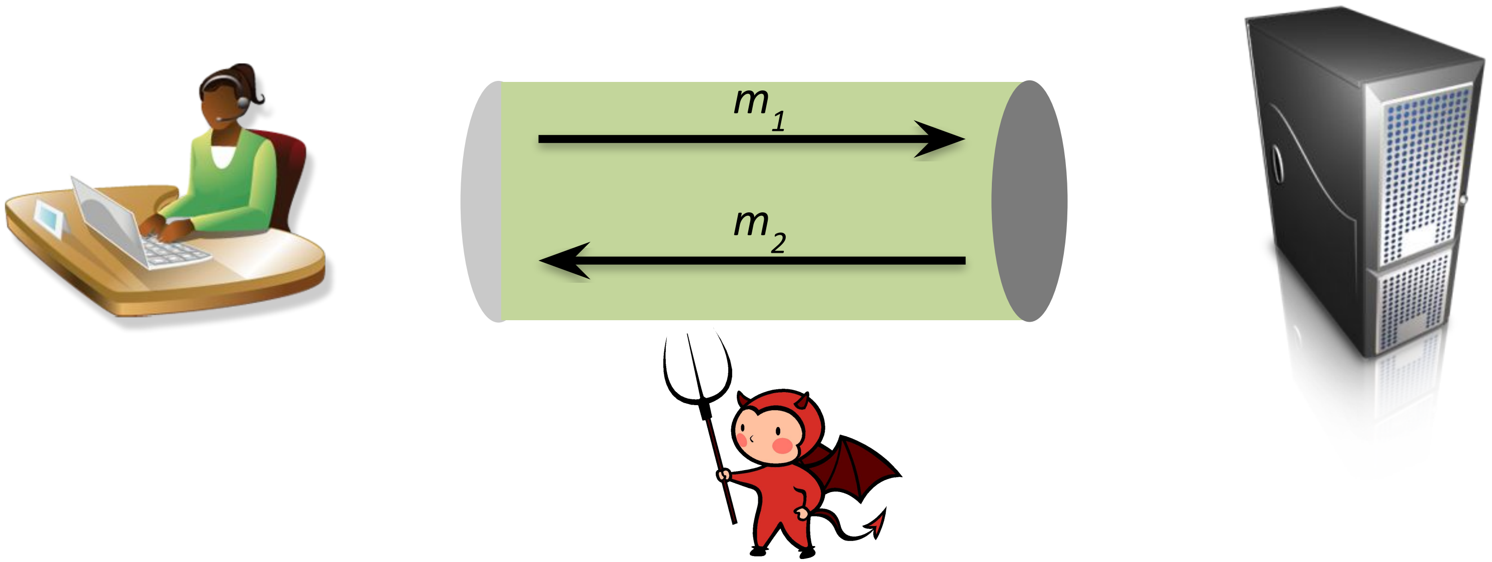
- What should our security goals be?
- What capabilities does **the adversary** (a.k.a. **the attacker**) have?
- How can we use cryptography to achieve our goals in the face of this adversary?

Secure communication: (informal) security goals



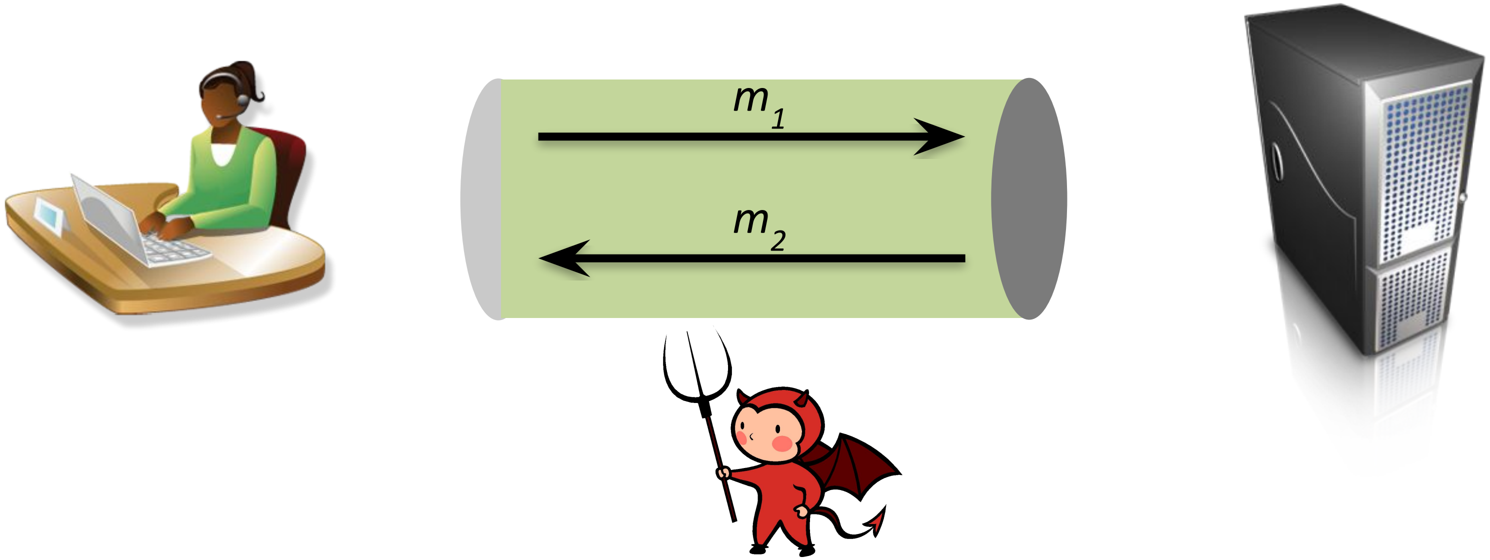
- **Exchanged messages should remain confidential.**
- Alice and Bob can check the **origin** of the messages (hard for the adversary to inject messages if its own).
- Alice and Bob can detect:
  - when messages are deleted.
  - when messages are reordered (possibly by the adversary, possibly by the network).

Secure communication: adversarial capabilities



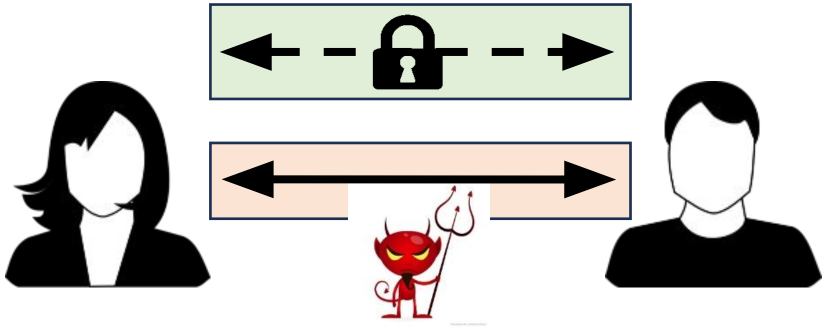
- **Passive adversary:**
  - Can only observe **all** of the data being transferred on the network.
- **Active adversary:**
  - Has sufficient control over the network to delete, delay, modify, and reorder network packets at will.
  - Can inject entirely new network packets (active adversary).
- **To what extent are these capabilities realistic?**

Secure communication: adversarial capabilities



- **In cryptography, we assume even more powerful adversaries**
  - Can ask for **chosen messages** to be passed over the network **(chosen plaintext attack)**.
  - Can observe effects of injecting **chosen network packets (chosen ciphertext attack)**.
    - For example, **error messages** exchanged between the entities in response to injected packets may leak useful information.

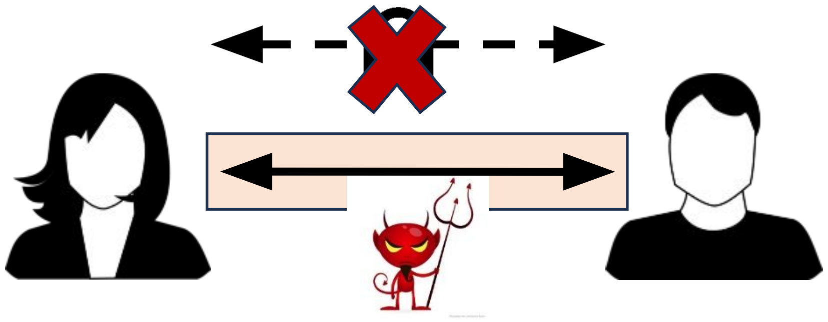
Secure communication: channel assumptions



Secure channel | Insecure channel

Symmetric-Key Cryptography

Assume a "costly" secure channel

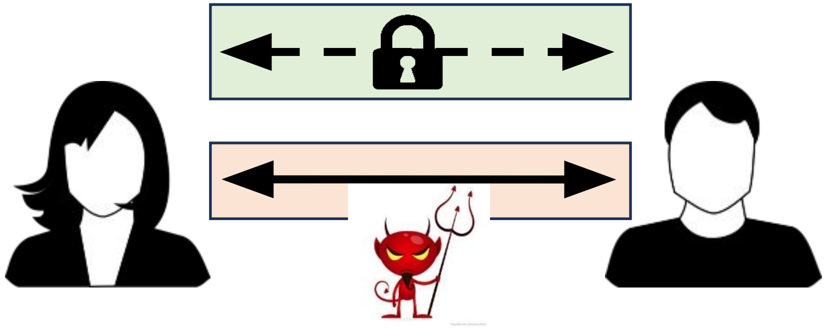


Insecure channel

Public-Key Cryptography

Do not assume a secure channel

Secure communication: coverage plan



Secure channel

Insecure channel

Symmetric-Key Cryptography

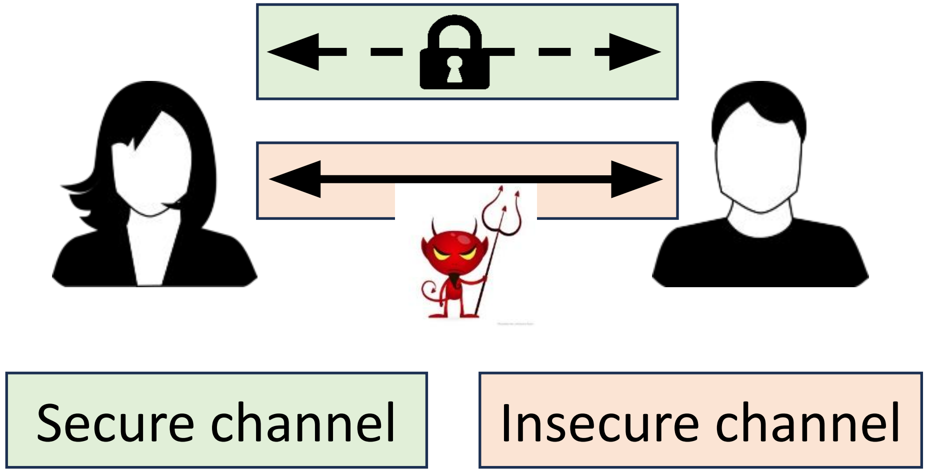
Assume a “costly” secure channel

For the next three sessions (rest of today and all of tomorrow):

- Assume a “costly” secure channel (can only be used to exchange “short” messages, albeit infrequently).
- Given this secure channel, design a secure channel for exchanging “arbitrarily long” messages very frequently.

Day-after tomorrow onwards: learn how to realize this costly secure channel

Secure communication: one-time pad



Symmetric-Key Cryptography

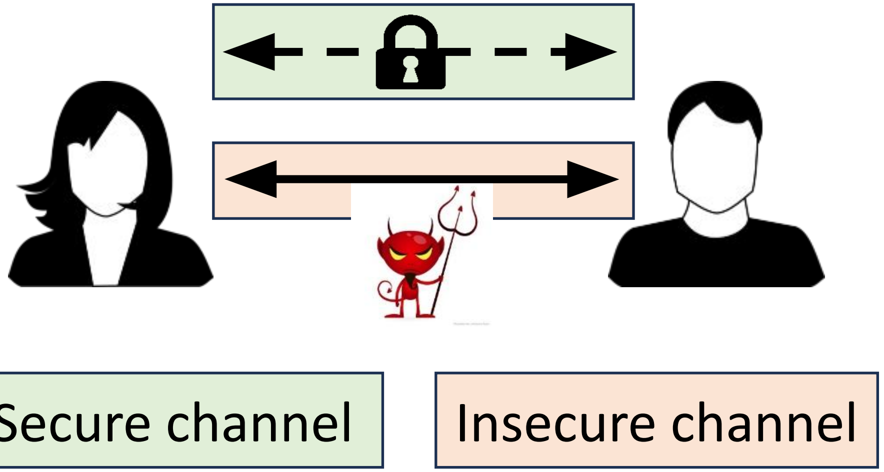
Assume a “costly” secure channel

One-time pad: perfectly secure, but…

- Need a secure channel to communicate the key (one-time pad) K
- K needs to be as long as the message
- K needs to be refreshed for each message to be communicated

Too costly to be practical

Secure communication: efficient one-time pad?



Symmetric-Key Cryptography

Assume a "costly" secure channel

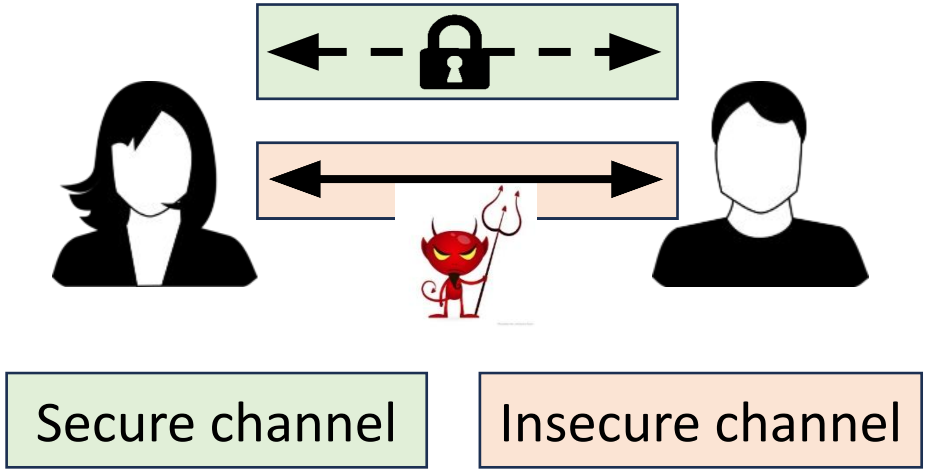
Practically efficient one-time pad

- A "short" key K is transmitted over the secure channel such that:
  - |K| is independent of message length.
  - K can be used to derive **arbitrarily many random bits**.
  - These random bits can then be used as **effective** one-time pads.

Too good to be true? Yes, for perfect security

Shannon [1949]

Secure communication: efficient one-time pad?



Symmetric-Key Cryptography

Assume a "costly" secure channel

Practically efficient one-time pad

- A "short" key K is transmitted over the secure channel such that:
  - |K| is independent of message length.
  - K can be used to derive **arbitrarily many random bits**.
  - These random bits can then be used as **effective** one-time pads.

But what if the adversary is **not all-powerful**, but **computationally bounded**?

Outline

- Some practical perspectives on cryptography
- Secure communication
- Computational cryptography – one-way functions
- Pseudorandom generators (PRGs) and stream ciphers

Computationally secure cryptography

Computational security

- Certain cryptosystems may not be perfectly/unconditionally/statistically secure against unbounded adversaries, but may still be “hard” to break in practice.
- **Computational security:** not unconditional, but holds against computationally bounded (equivalently, efficient) adversaries.
- Note that a (computationally bounded) adversary can always break a cryptosystem with some “tiny” probability (e.g., by guessing a key), so any meaningful notion of computational security is probabilistic.

Computational security: concrete formulation

Concrete formulation

- $(\boldsymbol{t}, \boldsymbol{\epsilon})$**-security:** a cryptosystem is said to satisfy (t, ϵ)-security if **any** adversary **running in time $\boldsymbol{t}$** fails to break it, **except with probability $\boldsymbol{\epsilon}$**

- Concrete formulation of computational security is mainstream in certain areas of cryptography, such as symmetric-key cryptography.

- Does not generalize very well (dependent on model of computation, tends to be cumbersome for theoretical analyses).

- Popular alternative: asymptotic formulation of computational security (next slide).

Computational security: asymptotic formulation

Asymptotic formulation (parameterized by security parameter λ)

A cryptosystem satisfies computational security w.r.t. **a security parameter $\boldsymbol{\lambda} \in \mathbb{N}$** if **any probabilistic polynomial-time (PPT)** adversary fails to break it, **except with probability $\mathbf{negl}(\boldsymbol{\lambda})$**.

Glossary

- $\text{poly}(n)$: a function $f(n) = \text{poly}(n)$ if $f(n) = n^{O(1)}$
- $\text{negl}(n)$: a function $f(n) = \text{negl}(n)$ if $f(n) = 1/n^{\omega(1)}$
- PPT: a randomized Turing machine with worst-case running time $\text{poly}(\lambda)$

One-way function (OWF)

Definition (informal)

- An OWF is an efficient, deterministic function that is efficiently computable but computationally hard to invert.

Definition (semi-formal)

- An OWF is a function $f: \{0,1\}^n \rightarrow \{0,1\}^m$ s.t. (for some security parameter λ):
  - There exists a PPT algorithm to compute $f(x)$ for any $x \in \{0,1\}^n$
  - Given $f(x)$ for $x \leftarrow_\$ \{0,1\}^n$, no PPT algorithm can compute x' s.t. $f(x') = f(x)$ except with probability $\mathrm{negl}(\lambda)$.

One-way function (OWF)

Definition (informal)

- An OWF is an efficient, deterministic function that is efficiently computable but computationally hard to invert.

Definition (formal)

- An OWF is a function $f: \{0,1\}^n \rightarrow \{0,1\}^m$ s.t. (for some security parameter λ):
  - There exists a PPT algorithm to compute $f(x)$ for any $x \in \{0,1\}^n$
  - For any $x \leftarrow_\$ \{0,1\}^n$ and any PPT algorithm A,

$$\Pr\left[A\left(1^\lambda, 1^n, 1^m, f(x)\right) = x' : f(x) = f(x')\right] \leq \mathrm{negl}(\lambda)$$

One-way function (OWF): Examples

Factorization

- A candidate OWF based on the hardness of integer factorization:

$$f: \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \rightarrow \{0,1\}^{2\lambda}, \text{ defined as } f(p, q) = pq$$

- Computationally efficient to multiply two λ-bit numbers.

- **Assumption:** No PPT algorithm can factorize $\mathrm{N} = pq$ when p and q are uniformly random λ-bit primes.

- Certain (implicit) assumptions about sampling primes and primality testing.

OWF: Complexity-theoretic perspective [Imp95]

- **Algorithmica:** $\mathrm{P} = \mathrm{NP}$ (or something "morally equivalent " such as $\mathrm{NP} \subseteq \mathrm{BPP}$)
- **Heuristica:** NP problems are hard in the worst case but easy on average.
- **Pessiland:** NP problems are hard on average but no one-way functions
- **Minicrypt:** One-way functions exist
- **Cryptomania:** Public-key cryptography exists

One-way permutation (OWP)

- An OWP is a function $f: \{0,1\}^n \to \{0,1\}^n$ s.t.
  - The function f is a bijection from $\{0,1\}^n$ to $\{0,1\}^n$.
  - The function f is an OWF.
- Examples: from number-theoretic assumptions (discrete log, RSA)
  - To be covered in future lectures.

Computational Indistinguishability

- Let $X = \{X_\lambda\}$ and $Y = \{Y_\lambda\}$ be two distribution ensembles over $\{0,1\}^{\ell(\lambda)}$ for $\ell(\lambda) = \text{poly}(\lambda)$.

- We say that X and Y are computationally indistinguishable if for any PPT distinguisher D we have $\left|\Pr\left[D\left(1^\lambda, X_\lambda\right) = 1\right] - \Pr\left[D\left(1^\lambda, Y_\lambda\right) = 1\right]\right| \leq \text{negl}(\lambda)$.

- This is sometimes summarized using the shorthand $X \approx_c Y$ (or simply, $X \approx Y$).

- **Reduction:** If $X = \{X_\lambda\}$ and $Y = \{Y_\lambda\}$ s.t. $X \approx_c Y$, then $\mathrm{f}(X) \approx_c f(Y)$ for any PPT function f.

- **Hybrid argument:** If $X = \{X_\lambda\}$, $Y = \{Y_\lambda\}$ and $Z = \{Z_\lambda\}$ s.t. $X \approx_c Y$ and $Y \approx_c Z$, then $X \approx_c Z$.

Outline

- Some practical perspectives on cryptography
- Secure communication
- Computational cryptography – one-way functions
- Pseudorandom generators (PRGs) and stream ciphers
- PRPs, PRFs, and block ciphers

Pseudorandom Generators (PRGs)

Efficient one-time pad

We will efficiently attempt to efficiently replicate the properties of the one-time pad:

- **One-time pad:** Key $K = K_0, K_1, K_2, \ldots$: a sequence of **random** bits
- **Our goal:** Keystream $K = K_0, K_1, K_2, \ldots$: a sequence of **pseudorandom** bits

Pseudorandom bits:

- Indistinguishable from random bits to a **computationally bounded adversary**.
- Generated cryptographically using a **Pseudorandom Generator (PRG)**.

Pseudorandom Generator (PRG)

Definition (informal)

- A PRG is an efficient, deterministic algorithm which takes as input a short "seed" and outputs a pseudorandom string.
- The output is usually longer than the input, and the added length is called the "stretch" of the generator.

Formal syntax

- A PRG is a function $G\colon \{0,1\}^{\ell} \to \{0,1\}^{L}$
- $\{0,1\}^{\ell}$ is called the **seed space**; $(L-\ell)$ is called the **stretch**.

Security of PRG

PRG $G : \{0,1\}^\ell \to \{0,1\}^L$

<u>Setup:</u>

$b \leftarrow_{\$} \{0,1\}$

Distinguisher D

r

<u>$b = 0$:</u>

$s \leftarrow_{\$} \{0,1\}^\ell$

$r = G(s)$

<u>$b = 1$:</u>

$r \leftarrow_{\$} \{0,1\}^L$

b'

$$\mathrm{Adv}_G^{\mathrm{PRG}}(D) := \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

Security of PRG

Definition (informal)

A PRG G is said to be **secure** if, for **all efficient** distinguishers D, the advantage

$$\mathrm{Adv}_G^{\mathrm{PRG}}(D) = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

is small.

- The power of the definition comes from the quantification over **all** D: this includes, e.g. all statistical tests, always outputting $b' = 0$, ...
- But still vague about what we mean by **efficient** and what we mean by **small**

Security of PRG

Definition (concrete)

A PRG G is said to be **(t, ϵ)-secure** if, for **all** distinguishers D **running in time at most t**

$$\mathrm{Adv}_G^{\mathrm{PRG}}(D) := \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \epsilon$$

Security of PRG

Definition (asymptotic)

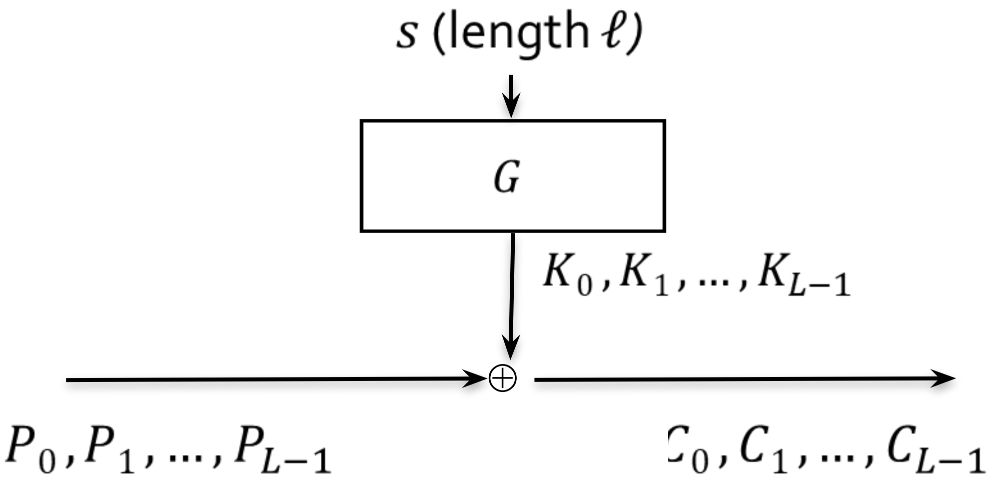
A PRG G is said to be **secure** if, for all security parameters $\lambda \in \mathbb{N}$ and **all** PPT distinguishers D

$$\mathrm{Adv}_G^{\mathrm{PRG}}(D) := \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \mathrm{negl}(\lambda)$$

Some results about PRGs (some proofs on the board)

- Given a PRG with 1-bit stretch, there exists a PRG with ℓ-bit stretch for $\ell = O(1)$
- Given a PRG with 1-bit stretch, there exists a PRG with $\ell(\lambda)$-bit stretch for $\ell(\lambda) = \mathrm{poly}(\lambda)$
- Given any OWP, there exists a PRG with 1-bit stretch [BluMic82,Yao82]
- Given any OWF, there exists a PRG with 1-bit stretch [HILL99]

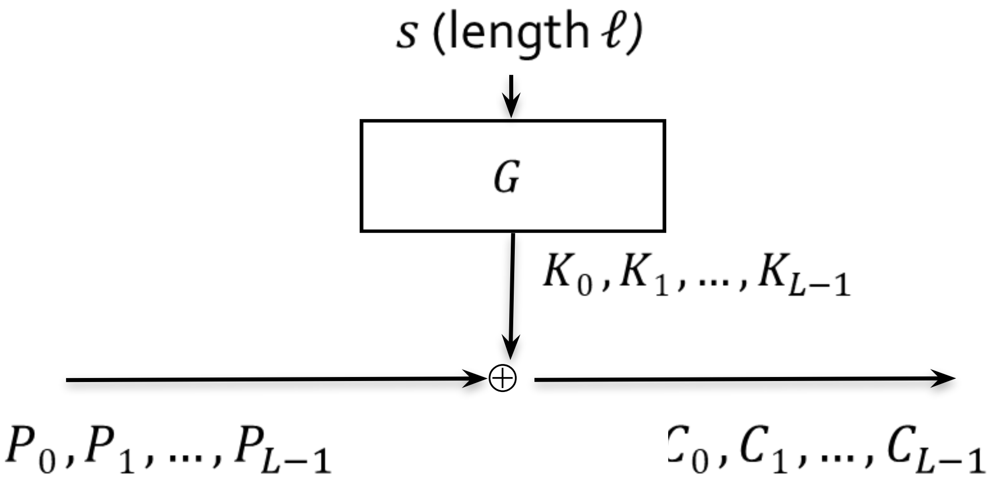
Using a PRG to realize efficient one-time pad



- Pros:
  - Cost-effective usage of a secure channel.
  - Instead of transmitting a long key (bit-length L) over the secure channel, only transmit a short seed (bit-length ℓ).

- **But** security is no longer perfect/unconditional, but only computational.

Using a PRG to realize efficient one-time pad



Board exercise: How to argue security of the efficient one-time pad?

PRGs in use: KeyStream Generators (KSGs)

Keystream Generators

Keystream Generators

A keystream generator (KSG) is an efficient, deterministic algorithm that:

- takes as input a seed s and an initialization vector IV, and
- outputs a stream of key bits $K = K_0, K_1, \ldots, K_L$

Formal syntax

- A KSG is a function H: $\{0,1\}^\ell \times \{0,1\}^v \rightarrow \{0,1\}^L$
- $\{0,1\}^\ell$ is called the **seed space**; $\{0,1\}^v$ is called the **initialization vector space**

IV is typically set to be a counter in applications (need not be secret for security of KSG)

Security of KSG

KSG $H\colon \{0,1\}^\ell \times \{0,1\}^v \to \{0,1\}^L$

Setup:

$b \leftarrow_\$ \{0,1\};\ s \leftarrow_\$ \{0,1\}^\ell$

Distinguisher D

$\mathrm{IV} \in \{0,1\}^v$

$r \in \{0,1\}^L$

b'

D makes up to q queries on ***distinct*** IVs of its choice.

$b = 0$:

$r = H(s, \mathrm{IV})$

$b = 1$:

$r \leftarrow_\$ \{0,1\}^L$

$$\mathrm{Adv}_H^{\mathrm{KSG}}(D) := \left|\Pr[b' = b] - \frac{1}{2}\right|.$$

Security of KSG

Definition (concrete)

A KSG H is said to be (q, t, ϵ)**-secure** if, for **all** distinguishers D **running in time at most** t and making **at most** q **queries** on **distinct** IVs

$$\mathrm{Adv}_H^{\mathrm{KSG}}(D) = \left|\Pr[b' = b] - \frac{1}{2}\right| \leq \epsilon$$

- Bit b and seed s are chosen once, queries made by the adversary are *adaptive*.
- Adversaries allowed to repeat IVs can trivially win the distinguishing game (why?)

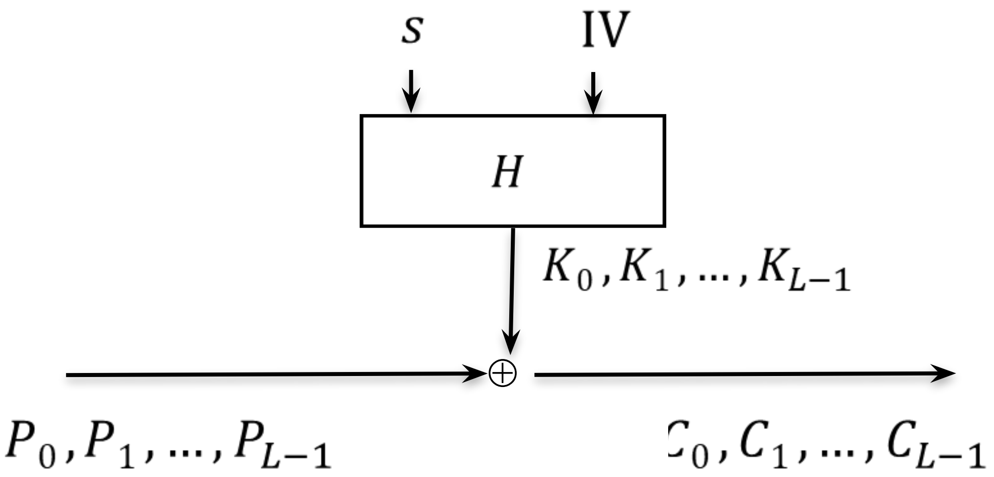
KSG from PRGs

- A KSG can be built from a PRG via careful design choices that combine the key and the seed of the KSG into the seed of the PRG.

- Bad design choices can lead to catastrophic security vulnerabilities:
  - Example: Wired Equivalent Privacy or WEP – an algorithm for 802.11 wireless networks, introduced as part of the IEEE 802.11 standard ratified in 1997).

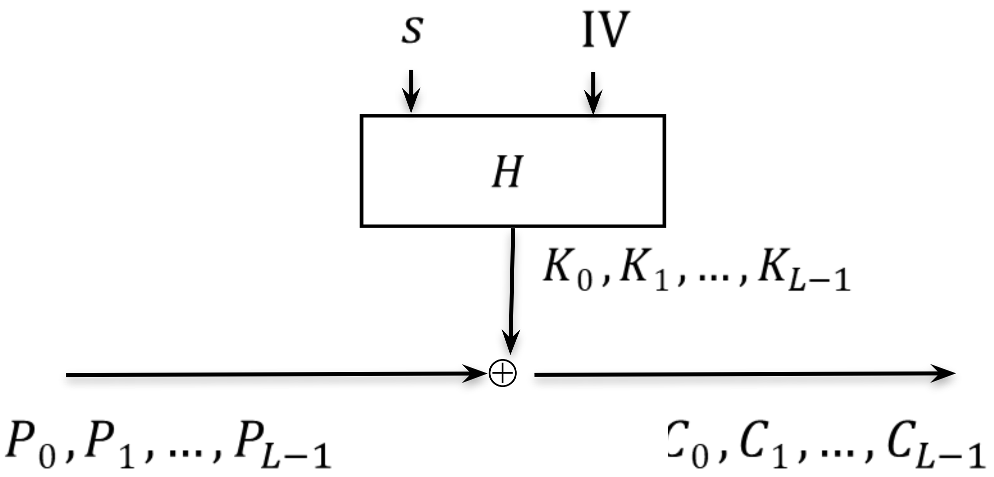
- In practice, use dedicated designs that mix key and IV together before producing the keystream.

Using a KSG to realize efficient one-time pad



- Pros:
  - Only transmit a short seed (bit-length ℓ).
  - IV can be sent publicly over the insecure channel along with $C = (C_0, \ldots, C_{L-1})$
- **Again**, security is not perfect/unconditional, but only computational.

Using a KSG to realize efficient one-time pad



Argument for security – similar to that for the PRG-based one-time pad

KSGs/PRGs in practice

Stream cipher

A stream cipher attempts to efficiently replicate the properties of the one-time pad:

- **One-time pad:** Key $K = K_0, K_1, K_2, \ldots$: a sequence of **random** bits
- **Stream cipher:** Keystream $K = K_0, K_1, K_2, \ldots$: a sequence of **pseudorandom** bits

Security (pseudorandom bits):

- Indistinguishable from random bits to a **computationally bounded adversary**.
- Based on practical variants of a PRG

Examples of stream ciphers: RC4

RC4 State

Byte permutation $\mathcal{S}$ and indices i and j

RC4 Key scheduling

```
begin
  for i = 0 to 255 do
    S[i] ← i
  end
  j ← 0
  for i = 0 to 255 do
    j ← j + S[i] + K[i mod keylen] mod 256
    swap(S[i], S[j])
  end
  i, j ← 0
end
```

RC4 Keystream generation

```
begin
  i ← i + 1 mod 256
  j ← j + S[i] mod 256
  swap(S[i], S[j])
  Z ← S[ S[i] + S[j] mod 256 ]
  return Z
end
```

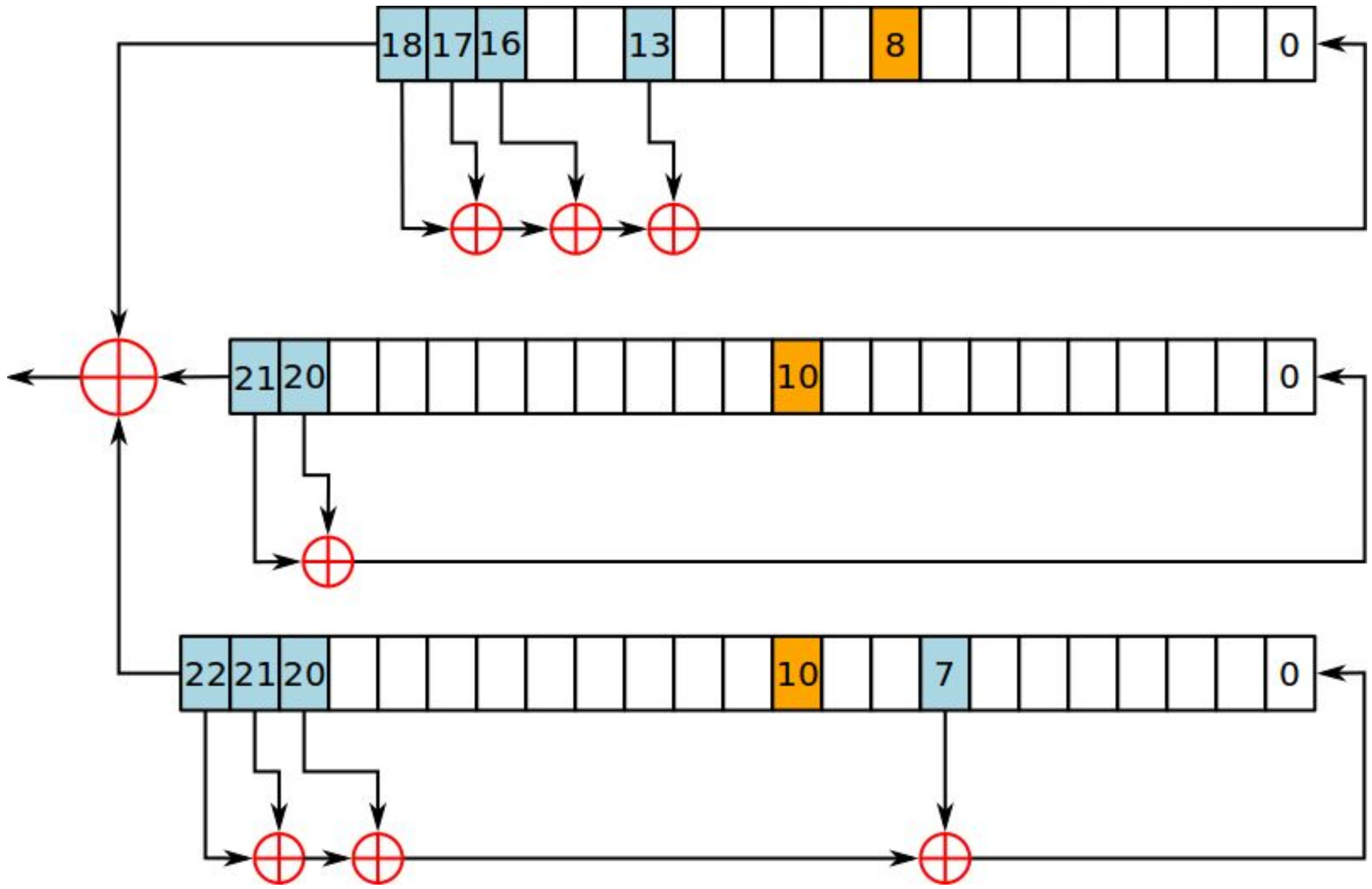
Examples of stream ciphers: RC4

- **Designed by Ron Rivest in late 1980s, became public in 1994.**
  - A byte-oriented algorithm with a variable-length key.
  - Elegant design, fast in software, very compact description, easy to implement in a few lines of ‘C’.
  - Heuristic realization of a PRG rather than a KSG – input is a key *K*, and there is no IV.

- **Became very widely adopted in secure communications protocols:**
  - TLS, WEP, WPA/TKIP, Kerberos.

RC4 has **serious security vulnerabilities** and is now **deprecated**

Examples of stream ciphers: A5/1



- Linear Feedback Shift Register (LFSR)-based design with stuttered clocking.
- Usage: 1980s till present day.
- Fast, low gate-count in hardware
  - Throughput: 114 bits/4.615ms
- Significant cryptanalysis.
  - Now considered insecure
  - Can recover key in a few seconds given a few hundred known plaintext bits.

Modern examples of stream ciphers

- Stream ciphers standardized by NIST (US):
    - AES in counter mode (in a few slides).

- Stream ciphers with IV identified by eSTREAM (EU project, 2008):
    - Profile 1 (“high throughput software applications”)
        - HC-128, Rabbit, Salsa20/12, SOSEMANUK
    - Profile 2 (“hardware applications with limited silicon area, power”)
        - Grain , MICKEY, Trivium

- Later development: ChaCha, a variant of Salsa, adopted by IETF for use in TLS.
    - Also, the default cipher in OpenSSH since release 6.8.
    - Also, used in Signal, Noise protocol framework, …

Security issues with stream ciphers in practice

- Keystream reuse
- Inherent weaknesses in generated keystream
- Complete lack of integrity checks

Security issues with stream ciphers in practice

- Keystream reuse
- Inherent weaknesses in generated keystream
- Complete lack of integrity checks

Keystream reuse

- Suppose plaintexts P_1, P_2 are encrypted with the same keystream K. So:

$$C_1 = P_1 \oplus K, \quad C_2 = P_2 \oplus K.$$

- Then, given C_1 and C_2, the adversary obtains: $C_1 \oplus C_2 = P_1 \oplus P_2$

- From $P_1 \oplus P_2$, it *may* be possible to learn the individual plaintexts P_1 and P_2.
  - Depends heavily on the plaintext distribution/statistics
  - Possible for natural language, application protocols such as HTTP, etc.
  - See Mason *et al*., "A Natural Language Approach to Automated Cryptanalysis of Two-time Pads" (https://www.cs.jhu.edu/~jason/papers/mason+al.ccs06.pdf).

Keystream reuse in practice: Example-1

- **Lorenz cipher**: A stream cipher operating on 5-bit teletype characters (used by German high command in WWII for wireless communications).

- **August 1941:** Operator error led to a repeated keystream
  - Revealed by repeated IV “HQIBPEXEZMUG”.
- **British recovered** $P_1 \oplus P_2$, then P_1 and P_2, then **the keystream** (John Tiltman), and eventually **reconstructed the entire Lorenz cipher** (William Tutte and others).
  - Developed some of the first electronic computers to industrialise the breaking of Lorenz (Tommy Flowers and the Colossus).
- … the British never saw an actual Lorenz machine.

Keystream reuse in practice: Example-2

- WEP (as used in WLANs) uses the RC4 stream cipher with a 40-bit seed s and a 24-bit initialization vector IV.
  - IV and seed are concatenated to make the actual RC4 seed: $s' = \mathrm{IV}\,||s$.
  - The seed is typically set by the user once and left fixed forever.
  - The IV is usually a counter (incremented for each frame sent on the network).

- Every 2^{24} frames, the seed and IV used as input to RC4 will repeat.
  - This results in repeated keystreams.
  - WEP has worse problems: combination of seed and IV by concatenation leads to correlations between seed bytes and output bytes.

Security issues with stream ciphers in practice

- Keystream reuse
- Inherent weaknesses in generated keystream
- Complete lack of integrity checks

Keystream biases: Example for RC4

- For many stream ciphers, keystreams can be efficiently distinguished from random in practice using simple statistical tests.
- **Example (RC4):**
  - **Mantin-Shamir (2001)**: Prob[second byte in RC4-generated keystream = 0x00] = 1/128 (ideally 1/256).
  - **Fluhrer-McGrew (2000)** and **Mantin (2005)**: *multibyte* biases in RC4-generated keystream.
  - **AlFardan-Bernstein-Paterson-Poettering-Schuldt (2013)**: for 128-bit seeds, all of the first 256 keystream bytes of RC4 are biased!
- These biases can be exploited to produce near-practical attacks on RC4 in various applications, including TLS and WPA/TKIP (the successor to WEP).
  - See for example https://www.usenix.org/conference/usenixsecurity13/technical-sessions/paper/alFardan

Security issues with stream ciphers in practice

- Keystream reuse
- Inherent weaknesses in generated keystream
- Complete lack of integrity checks

Stream ciphers do not provide integrity

- Stream ciphers provide no **integrity (authorization checks)** against active attackers.

- Flipping a bit ($0 \rightarrow 1$ or $1 \rightarrow 0$) in the ciphertext stream has the effect of producing a bit-flip in the plaintext stream in the same position:

$$C_i = P_i \oplus K_i \quad \Rightarrow \quad C_i \oplus 1 = (P_i \oplus 1) \oplus K_i$$

- Hence decryption algorithm will output $P_i \oplus 1$ instead of P_i.
  - Undetectable plaintext modifications by active adversary.
  - Highlights the general fact that encryption does **not** provide integrity.
  - Can lead to practical attacks on real systems (more later).