Definitions!

Advanced Topics in ~~Cybersecurity~~Cryptography (7CCSMATC)

Martin R. Albrecht

Outline

Introduction

Example 1: “What does it mean to generate random-looking bits?”

Example 2: “What does it mean for a block cipher to be secure?”

Example 3: “What does it mean for an encryption scheme to be secure?”

Exercise

Introduction

Main Reference

Phillip Rogaway. **On the role definitions in and beyond cryptography.** In: *Annual Asian Computing Science Conference*. Springer. 2004, pp. 13–32

- One of the most influential cryptographers in the world
- Many cryptographic solutions rely on definitions, models and constructions by him
- Fellow of the IACR
- Winner of the Levchin prize
- NSF CAREER Award in 1996[a]

[a] Which the NSA had attempted to prevent by influencing the NSF.

Motivation

kennyog @kennyog · Jan 9

We disclosed the vulnerabilities to @ThreemaApp back in October. Congrats to the team there on getting everything fixed quickly.

1 1 28 3,877

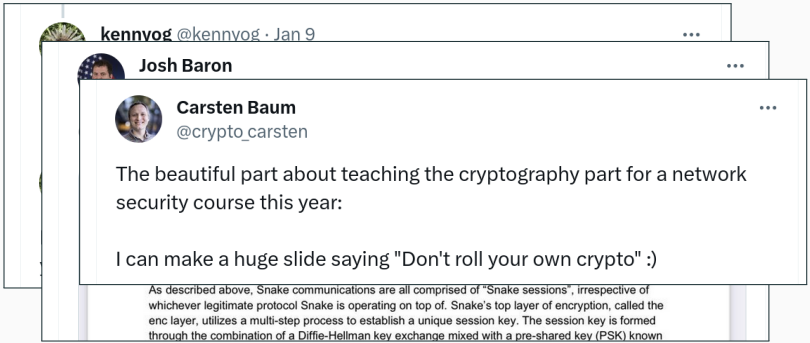
kennyog
@kennyog

Lessons learned 1: using a good crypto library is not enough - don't roll your own protocol.

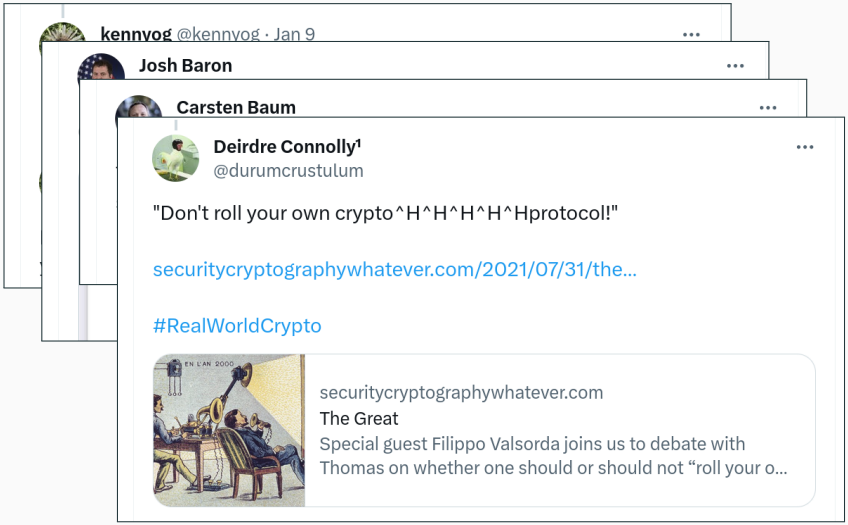
Motivation


kennyog @kennyog · Jan 9
Josh Baron
@JoshuaWBaron
Cryptography is hard: Russian malware edition
JOINT
CYBERSECURITY ADVISORY
TLP:CLEAR
International Partnership
Snake “enc” Layer
As described above, Snake communications are all comprised of “Snake sessions”, irrespective of whichever legitimate protocol Snake is operating on top of. Snake’s top layer of encryption, called the enc layer, utilizes a multi-step process to establish a unique session key. The session key is formed through the combination of a Diffie-Hellman key exchange mixed with a pre-shared key (PSK) known

Motivation


kennyog @kennyog · Jan 9
Josh Baron
Carsten Baum
@crypto_carsten
The beautiful part about teaching the cryptography part for a network security course this year:
I can make a huge slide saying "Don't roll your own crypto" :)
As described above, Snake communications are all comprised of "Snake sessions", irrespective of whichever legitimate protocol Snake is operating on top of. Snake's top layer of encryption, called the enc layer, utilizes a multi-step process to establish a unique session key. The session key is formed through the combination of a Diffie-Hellman key exchange mixed with a pre-shared key (PSK) known

Motivation


kennyog @kennyog · Jan 9
Josh Baron
Carsten Baum
Deirdre Connolly[1]
@durumcrustulum
"Don't roll your own crypto^H^H^H^H^Hprotocol!"
securitycryptographywhatever.com/2021/07/31/the...
#RealWorldCrypto
EN L'AN 2000
securitycryptographywhatever.com
The Great
Special guest Filippo Valsorda joins us to debate with Thomas on whether one should or should not “roll your o...

Motivation

"Crypto is hard!"

Motivation

"Crypto is hard!"

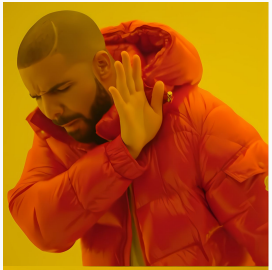
Of course, cryptography is hard, so is any other science.

Motivation

"Crypto is hard!"

Modern cryptography gives us the tools to reason about cryptographic protocols to rule out weaknesses.

Motivation



“Crypto is hard!”

“Cryptography needs security models and proofs!”

For the Record

You should neither design nor implement cryptographic primitives or protocols.[1]

[1] **You** should also neither design nor build bridges etc.

Definitions in cryptography emerged rather suddenly, in 1982, with the work of Shafi Goldwasser and Silvio Micali [GM84]. Before then, cryptography was all about schemes and attacks, and there was no way to gain confidence in a scheme beyond that which was had when smart people failed to find an attack.

*What Goldwasser and Micali did was, first of all, to **define** cryptography’s classical goal, message privacy (the goal of an **encryption scheme**).*
The [GM84] definition was strong and satisfying, and they proved it equivalent to very different-looking alternatives.
*Next they gave a **protocol** for encryption, and **proved** their protocol satisfied their definition, given a complexity-theoretic assumption.*
*The proof took the form of a **reduction**: if the encryption protocol didn’t meet the security definition, some other protocol wouldn’t satisfy its security definition.*
*The **definition–protocol–proof** approach has come to be called **provable security**.*

Shafi Goldwasser and Silvio Micali. **Probabilistic Encryption**. In: *Journal of Computer and System Sciences* 28.2 (1984), pp. 270–299[a]

[a] This work earned the Turing Award in 2012: “For transformative work that laid the complexity-theoretic foundations for the science of cryptography and in the process pioneered new methods for efficient verification of mathematical proofs in complexity theory”

Example 1: "What does it mean to generate random-looking bits?"

Impossibility?

- We aim to produce pseudorandom bits that are so much like the genuine article that no feasible program $\mathcal{D}$ can tell the difference.
- This might sound impossible.
- If we generate any fixed string of pseudorandom bits y, we have no chance:
  - For any y there is a simple algorithm $\mathcal{D}$ that does a good job at telling if it is given y or random bits.
  - What is that program?

```
int get_random_number() {
  return 4; // chosen by fair dice roll.
            // guaranteed to be random.
}
```

Credit: https://xkcd.com/221/

Take away

Our method of making pseudorandom bits must itself be randomised.

We therefore focus on an object, a **pseudorandom generator (PRG)**, that stretches a random **seed**, s, into a longer, pseudorandom string. Formally:

Definition (PRG Syntax)

A PRG is a function $G : \{0,1\}^n \to \{0,1\}^N$ where $N > n \geq 1$ are constants associated to G.

We therefore focus on an object, a **pseudorandom generator (PRG)**, that stretches a random **seed**, s, into a longer, pseudorandom string. Formally:

Definition (PRG Syntax)

A PRG is a function $G : \{0,1\}^n \to \{0,1\}^N$ where $N > n \geq 1$ are constants associated to G.

We are not done

This is just syntax: $G(s) = 0^N$ satisfies the API.

Security I

- We play a game between a challenger and a distinguisher $\mathcal{D}$ or adversary (often written $\mathcal{A}$).
- A distinguisher is just an algorithm, possibly a probabilistic one, equipped with a way to interact with its environment.
- We equip $\mathcal{D}$ with an oracle that has a "button" that $\mathcal{D}$ can push. Each time $\mathcal{D}$ pushes the button it gets a string of length N.
- The distinguisher outputs a bit $b \in \{0, 1\}$ encoding its answer.

We play a game:

- Game_0 every time the distinguisher $\mathcal{D}$ makes an oracle query, choose a random string $y \in \{0,1\}^N$ and return it.
- Game_1 every time the distinguisher $\mathcal{D}$ makes an oracle query, choose a random $x \in \{0,1\}^n$ and give out $y = G(x)$.

Game_0	
1 :	$b \leftarrow \mathcal{D}^{\text{G}}$
2 :	**return** b

Game_1	
1 :	$b \leftarrow \mathcal{D}^{\text{G}}$
2 :	**return** b

G()	
1 :	$y \leftarrow\!\!\$\ \{0,1\}^N$
2 :	$x \leftarrow\!\!\$\ \{0,1\}^n$ //Game_1
3 :	$y \leftarrow G(x)$ //Game_1
4 :	**return** y

Figure 1: PRG Security Games.

Security III

- The distinguisher wants to decide if it is playing Game_0 or Game_1.
  - So it outputs a bit b and then halts, where b matches the game it thinks it is playing.
- The advantage of $\mathcal{D}$ in attacking G is defined as

$$\text{Adv}^{\text{prg}}_G(\mathcal{D}) = \left| \Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1] \right|.$$

  - We write $\Pr[\mathcal{A}^{\text{Game}}]$ for the probability of $\mathcal{A}$ outputting one in the game Game.
  - We write $\Pr[\text{Game}^{\mathcal{A}}]$ for the probability of Game outputting one with adversary $\mathcal{A}$.
- For our bad generator $G(x) = 0^N$ there is an easy distinguisher: If the output is all zero, return 0, otherwise return 1.

Concrete Security

We say a PRG is secure if no "efficient" $\mathcal{D}$ exists:

$$\forall\, \mathcal{D} \in t \text{ steps}: \ \mathsf{Adv}^{\mathrm{prg}}_{G}(\mathcal{D}) = |\Pr[\mathcal{D}^{\mathsf{Game}_1} = 1] - \Pr[\mathcal{D}^{\mathsf{Game}_0} = 1]| < \varepsilon.$$

For example $(n, t, \varepsilon) = (128, 2^{64}, 2^{-64})$, s.t. $t/\varepsilon = 2^{128}$.

Concrete Security

We say a PRG is secure if no "efficient" $\mathcal{D}$ exists:

$$\forall\, \mathcal{D} \in t \text{ steps}: \ \mathrm{Adv}_G^{\mathrm{prg}}(\mathcal{D}) = |\Pr[\mathcal{D}^{\mathrm{Game}_1} = 1] - \Pr[\mathcal{D}^{\mathrm{Game}_0} = 1]| < \varepsilon.$$

For example $(n, t, \varepsilon) = (128, 2^{64}, 2^{-64})$, s.t. $t/\varepsilon = 2^{128}$.

- 2Ghz CPU: $\approx 2^{31} = 2 \cdot 2^{3 \cdot 10}$ ops per second
- $3600 \cdot 24 \cdot 365 \cdot 100 \approx 2^{11} \cdot 2^5 \cdot 2^8 \cdot 2^7 \approx 2^{31}$ seconds in 100 years
- 190 billion $\approx 2^{38}$ ARM chips by 2020.

All ARM cores ever clocked at 2GHz for 100 years gives you only 2^{100} ops.[a]

[a] All Bitcoin miners together perform about 2^{68} hashes per second.

Concrete Security

We say a PRG is secure if no "efficient" $\mathcal{D}$ exists:

$$\forall\, \mathcal{D} \in t \text{ steps}: \ \mathsf{Adv}_G^{\mathrm{prg}}(\mathcal{D}) = |\Pr[\mathcal{D}^{\mathsf{Game}_1} = 1] - \Pr[\mathcal{D}^{\mathsf{Game}_0} = 1]| < \varepsilon.$$

For example $(n, t, \varepsilon) = (128, 2^{64}, 2^{-64})$, s.t. $t/\varepsilon = 2^{128}$.

- 2Ghz CPU: $\approx 2^{31} = 2 \cdot 2^{3 \cdot 10}$ ops per second
- $3600 \cdot 24 \cdot 365 \cdot 100 \approx 2^{11} \cdot 2^5 \cdot 2^8 \cdot 2^7 \approx 2^{31}$ seconds in 100 years
- 190 billion $\approx 2^{38}$ ARM chips by 2020.

All ARM cores ever clocked at 2GHz for 100 years gives you only 2^{100} ops.[a]

- $(t, \varepsilon) = (2^{128}, 1)$?

[a] All Bitcoin miners together perform about 2^{68} hashes per second.

Concrete Security

We say a PRG is secure if no "efficient" $\mathcal{D}$ exists:

$$\forall\, \mathcal{D} \in t \text{ steps}: \ \mathsf{Adv}^{\mathrm{prg}}_{G}(\mathcal{D}) = |\Pr[\mathcal{D}^{\mathsf{Game}_1} = 1] - \Pr[\mathcal{D}^{\mathsf{Game}_0} = 1]| < \varepsilon.$$

For example $(n, t, \varepsilon) = (128, 2^{64}, 2^{-64})$, s.t. $t/\varepsilon = 2^{128}$.

- 2Ghz CPU: $\approx 2^{31} = 2 \cdot 2^{3 \cdot 10}$ ops per second
- $3600 \cdot 24 \cdot 365 \cdot 100 \approx 2^{11} \cdot 2^5 \cdot 2^8 \cdot 2^7 \approx 2^{31}$ seconds in 100 years
- 190 billion $\approx 2^{38}$ ARM chips by 2020.

All ARM cores ever clocked at 2GHz for 100 years gives you only 2^{100} ops.[a]

[a] All Bitcoin miners together perform about 2^{68} hashes per second.

- $(t, \varepsilon) = (2^{128}, 1)$?
  - There might be a fast algorithm that succeeds with probability 1/3.
  - Often can run our $(t, \varepsilon) = (2^{64}, 2^{-64})$ attacker 2^{64} times to get $\approx (2^{128}, 1)$.

Concrete Security

We say a PRG is secure if no "efficient" $\mathcal{D}$ exists:

$$\forall\, \mathcal{D} \in t \text{ steps}: \ \mathsf{Adv}_G^{\mathrm{prg}}(\mathcal{D}) = |\Pr[\mathcal{D}^{\mathsf{Game}_1} = 1] - \Pr[\mathcal{D}^{\mathsf{Game}_0} = 1]| < \varepsilon.$$

For example $(n, t, \varepsilon) = (128, 2^{64}, 2^{-64})$, s.t. $t/\varepsilon = 2^{128}$.

- 2Ghz CPU: $\approx 2^{31} = 2 \cdot 2^{3 \cdot 10}$ ops per second
- $3600 \cdot 24 \cdot 365 \cdot 100 \approx 2^{11} \cdot 2^5 \cdot 2^8 \cdot 2^7 \approx 2^{31}$ seconds in 100 years
- 190 billion $\approx 2^{38}$ ARM chips by 2020.

All ARM cores ever clocked at 2GHz for 100 years gives you only 2^{100} ops.[a]

[a] All Bitcoin miners together perform about 2^{68} hashes per second.

- $(t, \varepsilon) = (2^{128}, 1)$?
  - There might be a fast algorithm that succeeds with probability 1/3.
  - Often can run our $(t, \varepsilon) = (2^{64}, 2^{-64})$ attacker 2^{64} times to get $\approx (2^{128}, 1)$.
- $(t, \varepsilon) = (1, 2^{-128})$?

Concrete Security

We say a PRG is secure if no "efficient" $\mathcal{D}$ exists:

$$\forall\, \mathcal{D} \in t \text{ steps}: \ \mathsf{Adv}_G^{\mathrm{prg}}(\mathcal{D}) = |\Pr[\mathcal{D}^{\mathsf{Game}_1} = 1] - \Pr[\mathcal{D}^{\mathsf{Game}_0} = 1]| < \varepsilon.$$

For example $(n, t, \varepsilon) = (128, 2^{64}, 2^{-64})$, s.t. $t/\varepsilon = 2^{128}$.

- 2Ghz CPU: $\approx 2^{31} = 2 \cdot 2^{3 \cdot 10}$ ops per second
- $3600 \cdot 24 \cdot 365 \cdot 100 \approx 2^{11} \cdot 2^5 \cdot 2^8 \cdot 2^7 \approx 2^{31}$ seconds in 100 years
- 190 billion $\approx 2^{38}$ ARM chips by 2020.

All ARM cores ever clocked at 2GHz for 100 years gives you only 2^{100} ops.[a]

[a] All Bitcoin miners together perform about 2^{68} hashes per second.

- $(t, \varepsilon) = (2^{128}, 1)$?
  - There might be a fast algorithm that succeeds with probability 1/3.
  - Often can run our $(t, \varepsilon) = (2^{64}, 2^{-64})$ attacker 2^{64} times to get $\approx (2^{128}, 1)$.
- $(t, \varepsilon) = (1, 2^{-128})$?
  - There might be an algorithm that runs in two steps but succeeds with very high probability.

Asymptotic Security

We say a PRG is secure if no "efficient" $\mathcal{D}$ exists:

$$\forall \mathcal{D} \in \text{PPT: } \text{Adv}_G^{\text{prg}}(\mathcal{D}) = \left| \Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1] \right| < 1/\text{poly}(n).$$

poly(n) any polynomial in n, e.g. n^2, n^{1000}, n^c where c is a constant

PPT probabilistic polynomial-time, an algorithm running in any polynomial time that may make random choices.

On $< 1/\mathrm{poly}(n)$ aka $\in \mathrm{negl}(n)$

- We insist on advantage $< 1/\mathrm{poly}(n)$, which we write as $\in \mathrm{negl}(n)$, because that rules out a polynomial-time adversary amplifying the success probability.
- Let m be the number of attempts by $\mathcal{D}$ and let win_i be the event that it succeeded in attempt i.
- Let p be an upper bound on the success probability of $\mathcal{D}$. Then, overall we have

$$\Pr[\mathrm{win}] = \Pr\left[\bigvee_{i=1}^{m} \mathrm{win}_i\right] \leq \sum_{i=1}^{m} \Pr[\mathrm{win}_i] \leq m \cdot p,$$

  where the second last bound is from the "union bound".
- If $p \in \mathrm{negl}(n)$ then $\mathrm{poly}(n) \cdot \mathrm{negl}(n) \in \mathrm{negl}(n)$ because $n^{\omega(1)-c} \in n^{\omega(1)}$ for any constant c.[2]

[2] You can think of $\log(n)$ or $\log\log(n)$ as examples of functions $\in \omega(1)$.

Definition (PRG)

A PRG is a function $G : \{0,1\}^n \to \{0,1\}^N$ where $N > n \geq 1$ are constants associated to G. We say G is (t, ε)-secure if

$$\forall \mathcal{D} \in t \text{ steps: } \mathrm{Adv}^{\mathrm{prg}}_G(\mathcal{D}) = \left| \Pr[\mathcal{D}^{\mathrm{Game}_1} = 1] - \Pr[\mathcal{D}^{\mathrm{Game}_0} = 1] \right| < \varepsilon$$

for Game_0 and Game_1 defined in Fig. 1.

We say G is secure if $t = \mathrm{poly}(n)$ and $\varepsilon = 1/\mathrm{poly}(n)$.

Example 2: "What does it mean for a block cipher to be secure?"

Definition (Block Cipher)

A blockcipher is a function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ where

- $\mathcal{K}$ is a finite, nonempty set (the key space),
- $n \geq 1$ is a number (the blocksize) and
- $E_k(\cdot)$ is a permutation (on $\{0,1\}^n$) for each $k \in \mathcal{K}$.

When $y := E_k(x)$ we call x the plaintext block and y the ciphertext block.

We may also define $x := E_k^{-1}(y)$ s.t. $y = E_k(x)$. We may write $D_k(\cdot) := E_k^{-1}(\cdot)$.

Permutation For every $y \in \{0,1\}^n$ and every k there exists **one** x s.t. $y = E_k(x)$.

Example AES is **the** blockcipher.

Security

What do you think should be the definition for block cipher security?

Security: Wrong Definitions

"Attacker looking at the ciphertext does not learn the key."

Security: Wrong Definitions

"Attacker looking at the ciphertext does not learn the key."

```
def E(k, x):
    return x
```

Security: Wrong Definitions

"Attacker looking at the ciphertext does not learn the key."

```
def E(k, x):
    return x
```

"Adversary looking at the ciphertext does not learn the plaintext."

Security: Wrong Definitions

"Attacker looking at the ciphertext does not learn the key."

```
def E(k, x):
    return x
```

"Adversary looking at the ciphertext does not learn the plaintext."

```
def E(k, x):
    y = (AES128(k, x[0]), x[1])
    return y
```

Security: Wrong Definitions

"Attacker looking at the ciphertext does not learn the key."

```
def E(k, x):
    return x
```

"Adversary looking at the ciphertext does not learn the plaintext."

```
def E(k, x):
    y = (AES128(k, x[0]), x[1])
    return y
```

"Adversary looking at the ciphertext learns nothing about the plaintext."

Security: Wrong Definitions

"Attacker looking at the ciphertext does not learn the key."

```
def E(k, x):
    return x
```

"Adversary looking at the ciphertext does not learn the plaintext."

```
def E(k, x):
    y = (AES128(k, x[0]), x[1])
    return y
```

"Adversary looking at the ciphertext learns nothing about the plaintext."

$\Rightarrow |x| \in \{0,1\}^n$

Security: Wrong Definitions

"Adversary looking at the ciphertext learns nothing about the plaintext except its length."

Security: Wrong Definitions

"Adversary looking at the ciphertext learns nothing about the plaintext except its length."

1. Bob asks "Do you like pizza?" in the clear
2. Alice responds with $E_k(\texttt{YES!DEFINITELY!!})$
3. Bob asks "Do you want to commit all the crimes?" in the clear?
4. Alice responds $E_k(\texttt{YES!DEFINITELY!!})$

Adversary knows Alice responded with the same answer to the two questions.

Blockciphers are terrible encryption schemes

You know this result as "Don't use ECB mode".

Side Note

Definitions can be wrong.

Security: Pseudorandom Permutation (PRP)

Game_0 a random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ is chosen and when $\mathcal{D}$ asks its oracle a question x, we return $\pi(x)$.

Game_0	
1 :	//n-bit perm.
2 :	$\pi \leftarrow\!\$\ \mathcal{P}^n$
3 :	**return** $\mathcal{D}^{\text{P}}$

Game_1	
1 :	$k \leftarrow\!\$\ \mathcal{K}$
2 :	**return** $\mathcal{D}^{\text{P}}$

P(x)	
1 :	$y \leftarrow \pi(x)$ //Game_0
2 :	$y \leftarrow E_k(x)$ //Game_1
3 :	**return** y

Game_1 a random key $k \in \mathcal{K}$ is chosen and when $\mathcal{D}$ asks its oracle a question x we return $y = E_k(x)$.

At the end the distinguisher needs to output its guess for what world we live in:

$$\text{Adv}^{\text{prp}}_E(\mathcal{D}) = |\Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1]|.$$

Security: PRP (Nope!)

As it stands, we could not play this game. Why?

Security: PRP (Fixed)

Game_0	$\text{P}(x)$
1 : $\pi \leftarrow \emptyset$	1 : **if** $x \in \pi.\text{keys}$ **then**
2 : **return** $\mathcal{D}^{\text{P}}$	2 : $\quad y \leftarrow \pi[x]$
Game_1	3 : **else**
1 : $\pi \leftarrow \emptyset$; $k \leftarrow\$ \mathcal{K}$	4 : $\quad y \leftarrow\$ \{0,1\}^n \setminus \pi.\text{values}$
2 : **return** $\mathcal{D}^{\text{P}}$	5 : $\quad \pi[x] \leftarrow y$
	6 : $y \leftarrow E_k(x)$ //Game_1
	7 : **return** y

Figure 2: PRP Security Games.

The adversary will make at most polynomially many queries so the size of the table π will be polynomially sized.

$E_k(\cdot)$, BUT WHAT ABOUT $E_k^{-1}(\cdot)$

We covered that the adversary might be able to call $E_k(\cdot)$ but not $E_k^{-1}(\cdot)$.

Security: Strong Pseudorandom Permutation (SPRP)

Game_0	$\mathrm{P}(x)$	$\mathrm{P}^{-1}(y)$
1 : $\pi \leftarrow \emptyset$	1 : **if** $x \in \pi.\text{keys}$ **then**	1 : **if** $y \in \pi.\text{values}$ **then**
2 : **return** $\mathcal{D}^{\mathrm{P},\mathrm{P}^{-1}}$	2 :	2 : // Find the x s.t. $\pi[x] = y$
Game_1	3 : $y \leftarrow \pi[x]$	3 : $x \leftarrow \pi^{-1}[y]$
1 : $k \leftarrow\!\!\$\ \mathcal{K}$	4 : **else**	4 : **else**
2 : **return** $\mathcal{D}^{\mathrm{P},\mathrm{P}^{-1}}$	5 : $y \leftarrow\!\!\$\ \{0,1\}^n \setminus \pi.\text{values}$	5 : $x \leftarrow\!\!\$\ \{0,1\}^n \setminus \pi.\text{keys}$
	6 : $\pi[x] \leftarrow y$	6 : $\pi[x] \leftarrow y$
	7 : $y \leftarrow E_k(x)$ //Game_1	7 : $x \leftarrow E_k^{-1}(y)$ //Game_1
	8 : **return** y	8 : **return** x

Figure 3: SPRP Security Games.

$$\mathrm{Adv}_E^{\mathrm{sprp}}(\mathcal{D}) = |\Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1]|.$$

Definition (PRP)

A PRP is a keyed permutation $E_k : \{0,1\}^n \to \{0,1\}^n$ for $k \leftarrow\!\$\, \mathcal{K}$.
We say E is (t, ε)-secure **PRP** if for Game_0 and Game_1 defined in Fig. 2 we have:

$$\forall\, \mathcal{D} \in t \text{ steps: } \text{Adv}^{\text{prp}}_E(\mathcal{D}) = \left|\Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1]\right| < \varepsilon \ .$$

We say E is (t, ε)-secure **SPRP** if for Game_0 and Game_1 defined in Fig. 3 we have:

$$\forall\, \mathcal{D} \in t \text{ steps: } \text{Adv}^{\text{sprp}}_E(\mathcal{D}) = \left|\Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1]\right| < \varepsilon \ .$$

We say $E.$ is (prp-,sprp-)secure if $t = \text{poly}(n)$ and $\varepsilon = 1/\text{poly}(n)$.

- Ideally, we want $t/\varepsilon \approx 2^n$
- No PRP can be $t = 2^n,\ \varepsilon = 1/\text{poly}(n)$ secure, if $|\mathcal{K}| < 2^n$:

Definition (PRP)

A PRP is a keyed permutation $E_k : \{0,1\}^n \to \{0,1\}^n$ for $k \leftarrow\!\$\, \mathcal{K}$.
We say E is (t, ε)-secure **PRP** if for Game_0 and Game_1 defined in Fig. 2 we have:

$$\forall\, \mathcal{D} \in t \text{ steps: } \text{Adv}^{\text{prp}}_E(\mathcal{D}) = \left|\Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1]\right| < \varepsilon \ .$$

We say E is (t, ε)-secure **SPRP** if for Game_0 and Game_1 defined in Fig. 3 we have:

$$\forall\, \mathcal{D} \in t \text{ steps: } \text{Adv}^{\text{sprp}}_E(\mathcal{D}) = \left|\Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1]\right| < \varepsilon \ .$$

We say $E.$ is (prp-,sprp-)secure if $t = \text{poly}(n)$ and $\varepsilon = 1/\text{poly}(n)$.

- Ideally, we want $t/\varepsilon \approx 2^n$
- No PRP can be $t = 2^n$, $\varepsilon = 1/\text{poly}(n)$ secure, if $|\mathcal{K}| < 2^n$: Try all possible $k \in \mathcal{K}$; there are only $|\mathcal{K}|$ many, but there are $n! \approx \sqrt{2\,\pi\, n} \cdot (\frac{n}{e})^n \gg 2^n$ permutations.

This covers all intuitive, feasible definitions discussed above

- "Attacker looking at the ciphertext does not learn the key." this would allow distinguishing
- "Adversary looking at the ciphertext does not learn the plaintext." this would allow distinguishing
- "Adversary looking at the ciphertext leans nothing about the plaintext except its length." output is random (up to permutation), but calling it twice returns the same value twice.

A Punchline

The correct definition might be unintuitive.

Example 3: "What does it mean for an encryption scheme to be secure?"

Definition (Authenticated Encryption)

An AE-scheme is a pair of algorithms (E, D) where

- $E : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^*$ is a randomised algorithm,
- $D : \mathcal{K} \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ is a deterministic algorithm,
- set $\mathcal{K} \subseteq \{0,1\}^*$ is nonempty and finite,
- $|E_k(m)| = |m| + \tau$ for some constant τ and
- $D_k(c) = m$ whenever $c = E_k(m)$.

Algorithms E and D are called the encryption algorithm and the decryption algorithm, and strings k, m and c are called the key, plaintext, and ciphertext.

Security

What do you think encryption should provide?

Security: Privacy (IND-CPA)

"Indistinguishability under Chosen Plaintext Attacks"

Game_0	$\mathsf{E}(m_0, m_1)$
1 : $k \leftarrow\!\!\$\ \mathcal{K}$	1 : **if** $\lvert m_0\rvert \neq \lvert m_1\rvert$ **then**
2 : **return** $\mathcal{D}^{\mathsf{E}}$	2 : **return** $\perp$
Game_1	3 : $c \leftarrow\!\!\$\ E_k(m_0)$
1 : $k \leftarrow\!\!\$\ \mathcal{K}$	4 : $c \leftarrow\!\!\$\ E_k(m_1)$ //Game_1
2 : **return** $\mathcal{D}^{\mathsf{E}}$	5 : **return** c

$$\mathsf{Adv}_{E}^{\text{ind-cpa}}(\mathcal{D}) = \left|\Pr[\mathcal{D}^{\mathsf{Game}_1} = 1] - \Pr[\mathcal{D}^{\mathsf{Game}_0} = 1]\right|$$

Security: Privacy (IND-CPA) Neater

IND-CPA		E(m_0, m_1)	
1 :	$k \leftarrow\!\$\ \mathcal{K}$	1 :	**if** $\lvert m_0\rvert \neq \lvert m_1\rvert$ **then**
2 :	$b \leftarrow\!\$\ \{0,1\}$	2 :	**return** $\perp$
3 :	**return** $\mathcal{D}^{\mathsf{E}}$	3 :	$c \leftarrow\!\$\ E_k(m_b)$
		4 :	**return** c

$$\mathsf{Adv}^{\text{ind-cpa}}_{E}(\mathcal{D}) = \left|\, \Pr[\mathcal{D}^{\text{IND-CPA}} = 1 | b = 1] - \Pr[\mathcal{D}^{\text{IND-CPA}} = 1 | b = 0] \right|$$

Security: Privacy (IND-CPA) Neatest

IND-CPA	E(m_0, m_1)
1 : $k \leftarrow\!\$\ \mathcal{K}$	1 : **if** $\lvert m_0\rvert \neq \lvert m_1\rvert$ **then**
2 : $b \leftarrow\!\$\ \{0,1\}$	2 : **return** $\perp$
3 : $b' \leftarrow \mathcal{D}^{\mathsf{E}}$	3 : $c \leftarrow\!\$\ E_k(m_b)$
4 : **return** $b = b'$	4 : **return** c

Figure 4: IND-CPA Security Game.

$$\mathsf{Adv}_{E}^{\text{ind-cpa}}(\mathcal{D}) = \left|\, \Pr[\text{IND-CPA}^{\mathcal{D}} = 1] - 1/2 \right|$$

Note: A "bad" adversary that always outputs $\neg b$ can be turned into a really good adversary always outputting b.

What does IND-CPA get you?

```python
from Crypto.Cipher import AES
from io import StringIO
import gzip

cipher = AES.new(k, AES.MODE_GCM)

def compress(m):
    out = StringIO()
    with gzip.GzipFile(fileobj=out, mode="w") as f:
        f.write(m)
    return out.getvalue()

def encrypt(k, m): # IND-CPA secure
    c, t = cipher.encrypt_and_digest(m)
    return c, t
```

```python
encrypt2 = lambda(k, m): encrypt(k, compress(m))
```

Wikipedia
The Free Encyclopedia

CRIME

6 languages

Article Talk Tools

From Wikipedia, the free encyclopedia

For criminal activity, see Crime. For other uses, see Crime (disambiguation).

CRIME (**Compression Ratio Info-leak Made Easy**) is a security vulnerability in HTTPS and SPDY protocols that utilize compression, which can leak the content of secret web cookies.[1] When used to recover the content of secret authentication cookies, it allows an attacker to perform session hijacking on an authenticated web session, allowing the launching of further attacks. CRIME was assigned CVE-2012-4929.[2]

Details [edit]

The vulnerability exploited is a combination of chosen plaintext attack and inadvertent information leakage through data compression, similar to that described in 2002 by the cryptographer John Kelsey.[3] It relies on the attacker being able to observe the size of the ciphertext sent by the browser while at the same time inducing the browser to make multiple carefully crafted web connections to the target site. The attacker then

Security: Integrity (INT-CTXT)

"Integrity of Ciphertexts"

INT-CTXT	E(m)
1 : $\mathcal{C} \leftarrow \emptyset$; $k \leftarrow\!\$\ \mathcal{K}$	1 : $c \leftarrow\!\$\ E_k(m)$
2 : $c \leftarrow \mathcal{A}^{\mathsf{E}}$	2 : $\mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$
3 : $m \leftarrow D_k(c)$	3 : **return** c
4 : **if** $c \notin \mathcal{C}$ and $m \neq \perp$ **then return** 1	
5 : **else return** 0	

Figure 5: INT-CTXT Security Game.

$$\mathsf{Adv}_E^{\text{int-ctxt}}(\mathcal{A}) = \Pr[\text{INT-CTXT}^{\mathcal{A}} = 1].$$

Security: Integrity (INT-CTXT) Neater

INT-CTXT	E(m)	D(c)
1 : $\mathcal{C} \leftarrow \emptyset$	1 : $c \leftarrow\!\$\; E_k(m)$	1 : **if** $b = 1$ **then**
2 : $k \leftarrow\!\$\; \mathcal{K}$	2 : $\mathcal{C} \leftarrow \mathcal{C} \cup \{(m, c)\}$	2 : **return** $D_k(c)$
3 : $b \leftarrow\!\$\; \{0,1\}$	3 : **return** c	3 : **else if** $\exists\,(c, m) \in \mathcal{C}$ **then**
4 : $b' \leftarrow \mathcal{A}^{\mathrm{E},\mathrm{D}}$		4 : **return** m
5 : **return** $b = b'$		5 : **else return** $\perp$

$$\mathrm{Adv}^{\text{int-ctxt}}_{E,D}(\mathcal{D}) = \left|\, \Pr[\text{INT-CTXT}^{\mathcal{D}} = 1] - 1/2 \right|.$$

Didn't we forget something?

- We defined PRP and SPRP security for block ciphers.

Didn't we forget something?

- We defined PRP and SPRP security for block ciphers.
- Our IND-CPA encryption scheme only provides an encryption oracle.
- Should we not also provide a decryption oracle?

IND-CCA

"Indistinguishability under Chosen Ciphertext Attacks"

IND-CCA	$\mathrm{E}(m_0, m_1)$	$\mathrm{D}(c)$
1 : $\mathcal{C} \leftarrow \emptyset$	1 : **if** $\lvert m_0\rvert \neq \lvert m_1\rvert$ **then**	1 : **if** $c \in \mathcal{C}$ **then**
2 : $k \leftarrow\$ \mathcal{K}$	2 : **return** $\perp$	2 : **return** $\perp$
3 : $b \leftarrow\$ \{0,1\}$	3 : $c \leftarrow\$ E_k(m_b)$	3 : **return** $D_k(c)$
4 : $b' \leftarrow \mathcal{D}^{\mathrm{E},\mathrm{D}}$	4 : **if** $m_0 \neq m_1$ **then** $\mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$	
5 : **return** $b = b'$	5 : **return** c	

Figure 6: IND-CCA Security Game.

$$\mathrm{Adv}^{\text{ind-cca}}_{E,D}(\mathcal{D}) = \left|\, \Pr[\text{IND-CCA}^{\mathcal{D}} = 1] - 1/2 \right|.$$

IND-CPA + INT-CTXT $\Rightarrow$ IND-CCA

Claim: If you have IND-CPA security and INT-CTXT security then you have IND-CCA security.

Argument Sketch: Simulate the decryption oracle!

Game_0	$\mathsf{E}(m_0, m_1)$	$\mathsf{D}(c)$
$T \leftarrow \emptyset$	**if** $\lvert m_0 \rvert \neq \lvert m_1 \rvert$ **then**	**if** $c \in T.\mathsf{keys}$ **then**
$k \leftarrow\!\$\ \mathcal{K}$	**return** $\perp$	**return** $T[c]$
$b \leftarrow\!\$\ \{0,1\}$	$c \leftarrow\!\$\ E_k(m_b)$	**return** $\perp$
$b' \leftarrow \mathcal{D}^{\mathsf{E},\mathsf{D}}$	**if** $m_0 = m_1$ **then** $T[c] \leftarrow m_b$	
return $b = b'$	**return** c	

Either IND-CCA adversary $\mathcal{D}$ submits some $c \notin T.\mathsf{keys}$ to D that decrypts correctly (breaking INT-CTXT) or it did not (breaking IND-CPA).

No Confidentiality without Integrity

CIA

The universally taught “CIA triad” is wrong: there is no cryptographic confidentiality without integrity.

“Having definitions makes it easier to come up with attacks.”

> *[T]he U.S. National Security Agency (NSA) released, through NIST, its own scheme for efficient AE. The mechanism was called* ***dual counter mode****. I myself read the definition of the mode and broke it in less than a day, privately informing NIST. Donescu, Gligor, and Wagner likewise broke the scheme right away [DGW01]. What is it that we knew that the folks from the super-secret NSA did not? The answer, I believe, is* ***definitions****. If you understood the definition for AE, it was pretty obvious that the NSA scheme wouldn’t work. In the absence of understanding a definition, you wouldn’t see it. Definitions, not initially intended to help people find attacks, nonetheless do just that.*

Pompiliu Donescu, Virgil D Gligor, and David Wagner. A note on NSA’s Dual Counter Mode of encryption. `https://people.eecs.berkeley.edu/~daw/papers/dcm-prelim.pdf`. 2001

Exercise

Write down the definition of a pseudorandom function (PRF)

Write down the definition of a pseudorandom function (PRF)

Definition (PRF)

A PRF is a keyed function $F_k : \{0,1\}^\lambda \to \{0,1\}^N$ where N depends on λ and for $k \leftarrow\!\$\, \mathcal{K}$. We say F_k is (t, ε)-secure **PRF** if for Game_0 and Game_1 defined below we have:

$$\forall\, \mathcal{D} \in t \text{ steps: } \text{Adv}_F^{\text{prf}}(\mathcal{D}) = \left|\Pr[\mathcal{D}^{\text{Game}_1} = 1] - \Pr[\mathcal{D}^{\text{Game}_0} = 1]\right| < \varepsilon$$

Game_0

1 : $f \leftarrow \emptyset$

2 : **return** $\mathcal{D}^{\text{F}}$

Game_1

1 : $f \leftarrow \emptyset$; $k \leftarrow\!\$\, \mathcal{K}$

2 : **return** $\mathcal{D}^{\text{F}}$

F(x)

1 : **if** $x \notin f$.keys **then** $f[x] \leftarrow\!\$\, \{0,1\}^N$

2 : $y \leftarrow f[x]$

3 : $y \leftarrow F_k(x)$ //Game_1

4 : **return** y

Say “no” to empty abstractions

“This cryptographic scheme is secure.”

“This cryptographic scheme achieves IND-CCA security, which means…”

References I

[DGW01] Pompiliu Donescu, Virgil D Gligor, and David Wagner. A note on NSA's Dual Counter Mode of encryption. https://people.eecs.berkeley.edu/~daw/papers/dcm-prelim.pdf. 2001.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. In: *Journal of Computer and System Sciences* 28.2 (1984), pp. 270–299.

[Rog04] Phillip Rogaway. On the role definitions in and beyond cryptography. In: *Annual Asian Computing Science Conference*. Springer. 2004, pp. 13–32.