



Four Attacks and a Proof for Telegram

Joint Latvian-Estonian Theory Days 2022
May 06, 2022

Martin R. Albrecht, Lenka Mareková, Kenneth G. Paterson, Igors Stepanovs



ETH zürich







Based on a paper to appear at IEEE S&P 2022.
More information at <https://mtpsym.github.io/>



Telegram

Monthly active users in Jan 2022:







According to Statista 2022.

	WhatsApp	$2000 \cdot 10^6$
	WeChat	$1263 \cdot 10^6$
	FB Messenger	$988 \cdot 10^6$
	QQ	$574 \cdot 10^6$
	Snapchat	$557 \cdot 10^6$
	Telegram	$550 \cdot 10^6$

Telegram

Monthly active users in Jan 2022:

According to Statista 2022.

	WhatsApp	$2000 \cdot 10^6$
	WeChat	$1263 \cdot 10^6$
	FB Messenger	$988 \cdot 10^6$
	QQ	$574 \cdot 10^6$
	Snapchat	$557 \cdot 10^6$
	Telegram	$550 \cdot 10^6$

Collective Information Security in Large-Scale Urban Protests: the Case of Hong Kong







Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and
Lenka Mareková, *Royal Holloway, University of London*

Predominant in Hong Kong protests.
Perceived more secure than competitors.

Telegram

Monthly active users in Jan 2022:

According to Statista 2022.

	WhatsApp	$2000 \cdot 10^6$
	WeChat	$1263 \cdot 10^6$
	FB Messenger	$988 \cdot 10^6$
	QQ	$574 \cdot 10^6$
	Snapchat	$557 \cdot 10^6$
	Telegram	$550 \cdot 10^6$

Collective Information Security in Large-Scale Urban Protests: the Case of Hong Kong

Martin R. Albrecht, Jorge Blasco, Rikke Bjerg Jensen, and Lenka Mareková, *Royal Holloway, University of London*

Predominant in Hong Kong protests.
Perceived more secure than competitors.

Advantages of Telegram:

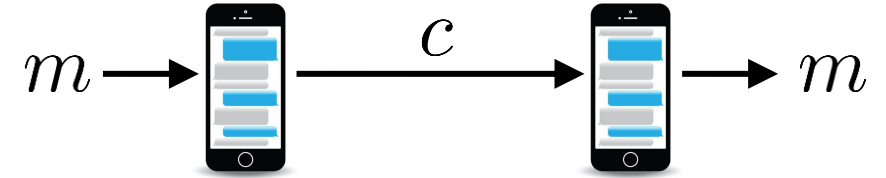
Group chats for up to 200000 people.
Support of pseudonyms in group chats.
Other features:

..., anonymous polls, disappearing messages, timed or scheduled messages, ability to delete messages sent by others, ...

Cloud Chats



Secret Chats

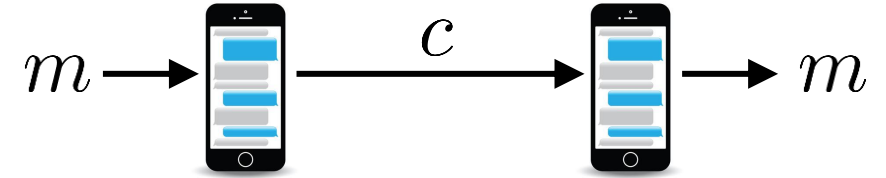


	Cloud Chats	Secret Chats
ENCRYPTION	client-server	end-to-end
GROUPS	✓	✗
1-ON-1	✓	✓
ENABLED BY DEFAULT	✓	✗

Cloud Chats



Secret Chats



	Cloud Chats	Secret Chats
ENCRYPTION	client-server	end-to-end
GROUPS	✓	✗
1-ON-1	✓	✓
ENABLED BY DEFAULT	✓	✗

MTProto protocol

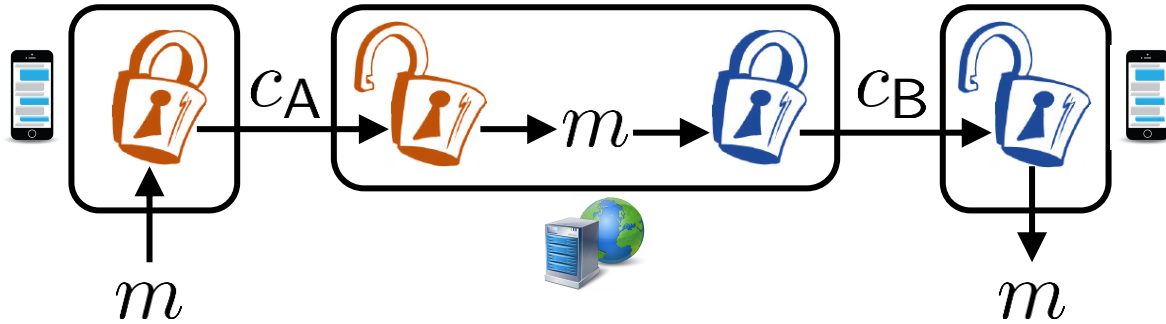


Encryption

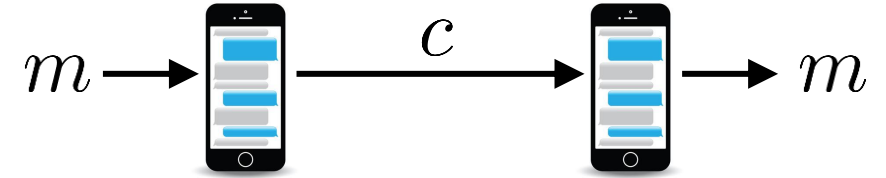


Decryption

Cloud Chats



Secret Chats



	Cloud Chats	Secret Chats
ENCRYPTION	client-server	end-to-end
GROUPS	✓	✗
1-ON-1	✓	✓
ENABLED BY DEFAULT	✓	✗

MTProto protocol

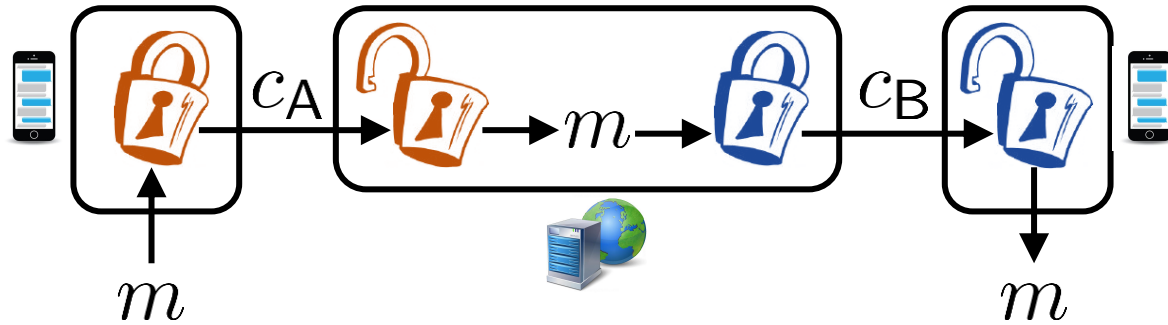


Encryption



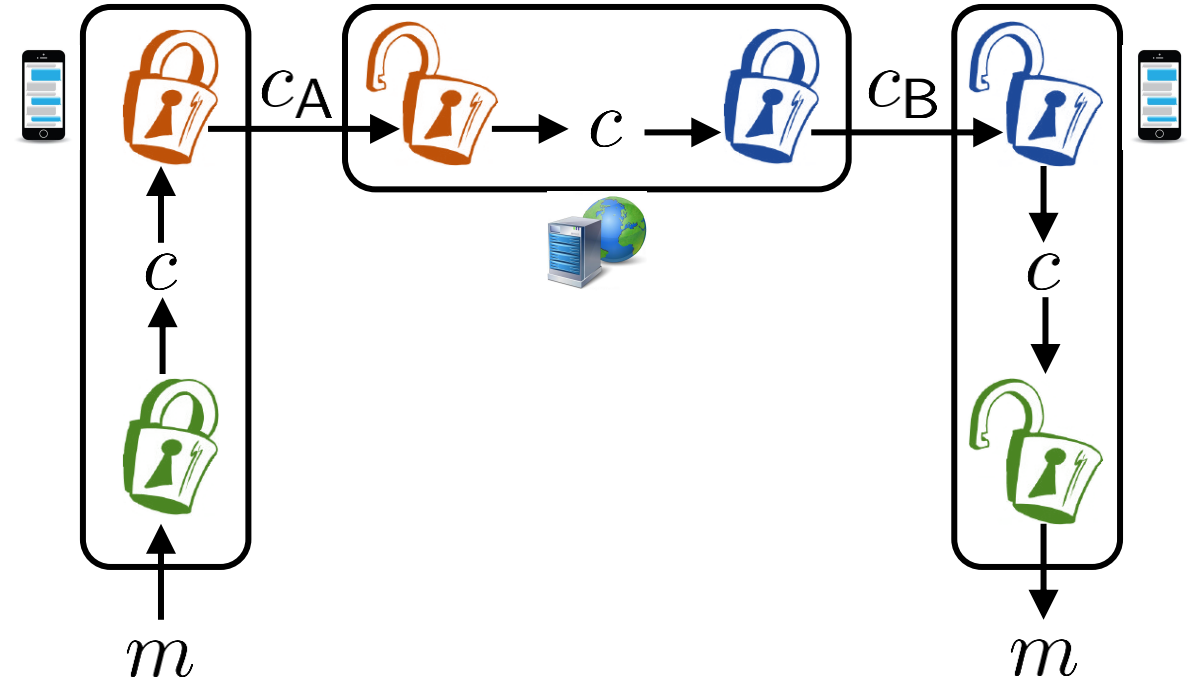
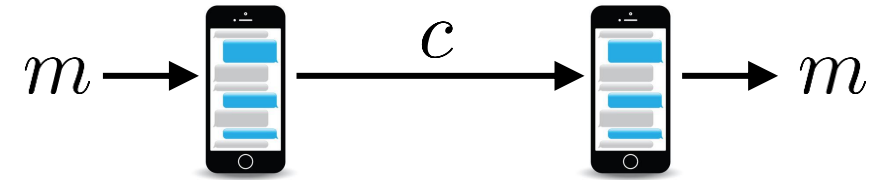
Decryption

Cloud Chats



	Cloud Chats	Secret Chats
ENCRYPTION	client-server	end-to-end
GROUPS	✓	✗
1-ON-1	✓	✓
ENABLED BY DEFAULT	✓	✗

Secret Chats



MTPProto protocol

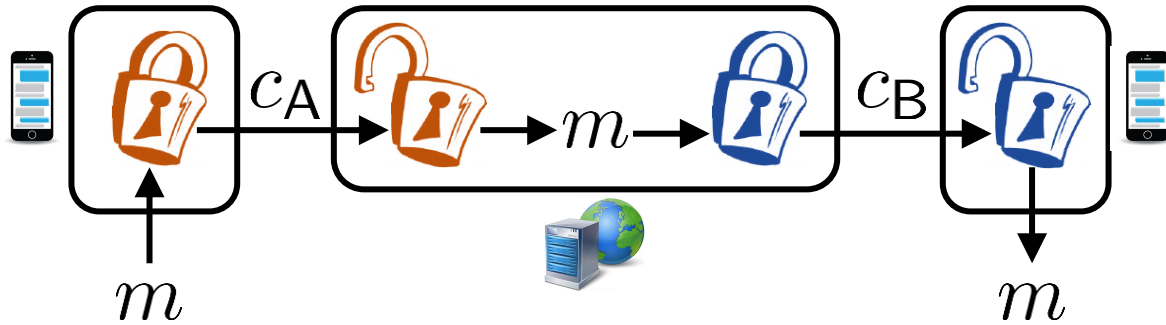


Encryption



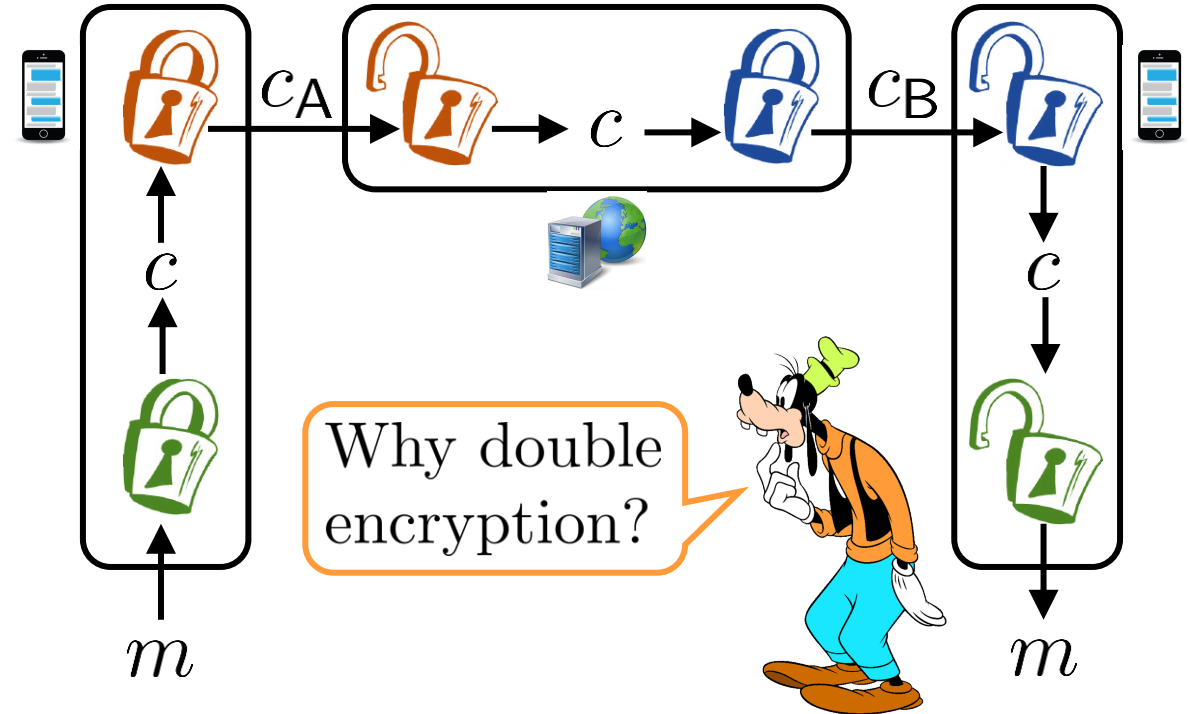
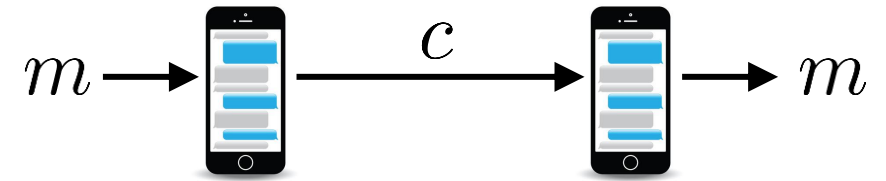
Decryption

Cloud Chats



	Cloud Chats	Secret Chats
ENCRYPTION	client-server	end-to-end
GROUPS	✓	✗
1-ON-1	✓	✓
ENABLED BY DEFAULT	✓	✗

Secret Chats



MTPProto protocol

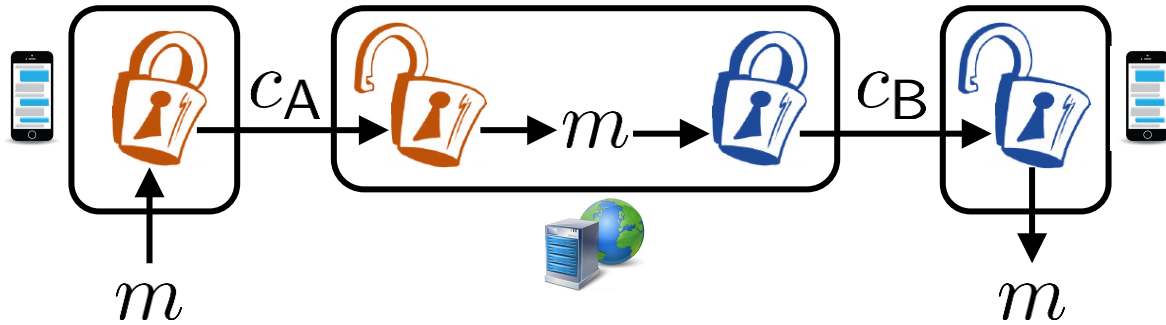


Encryption



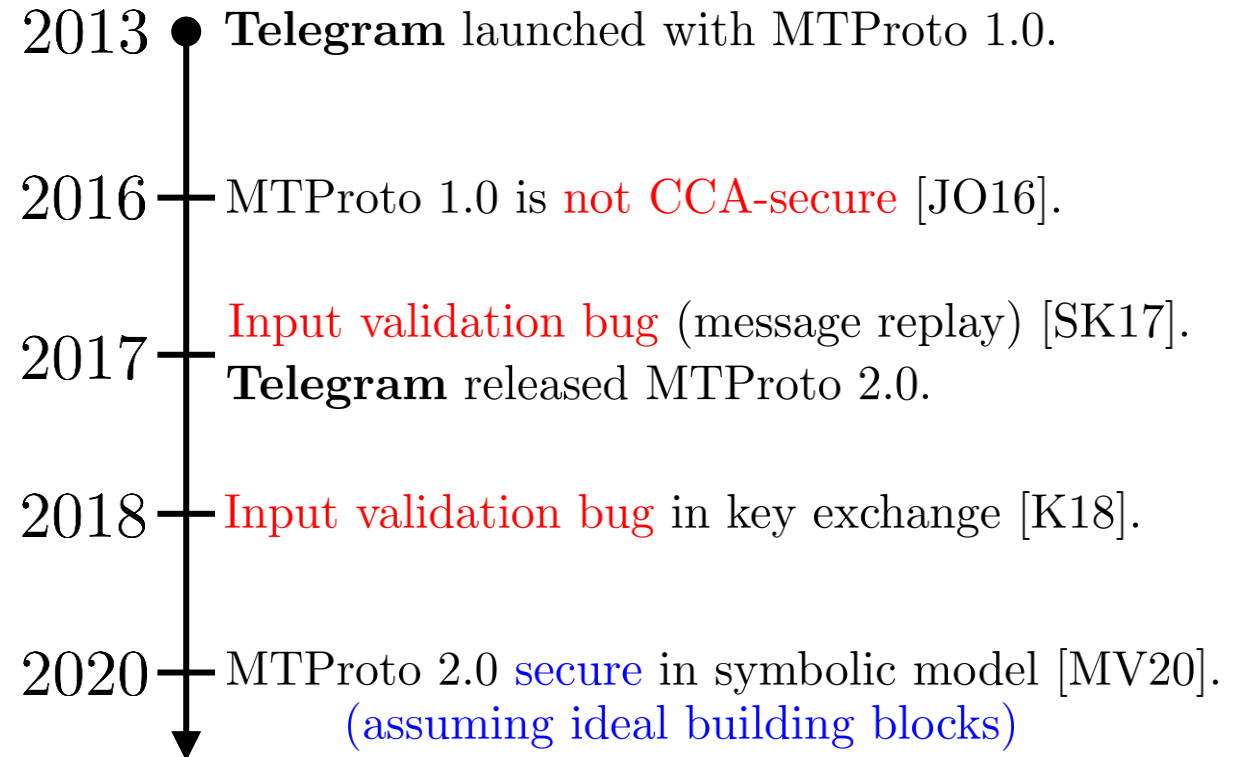
Decryption

Cloud Chats



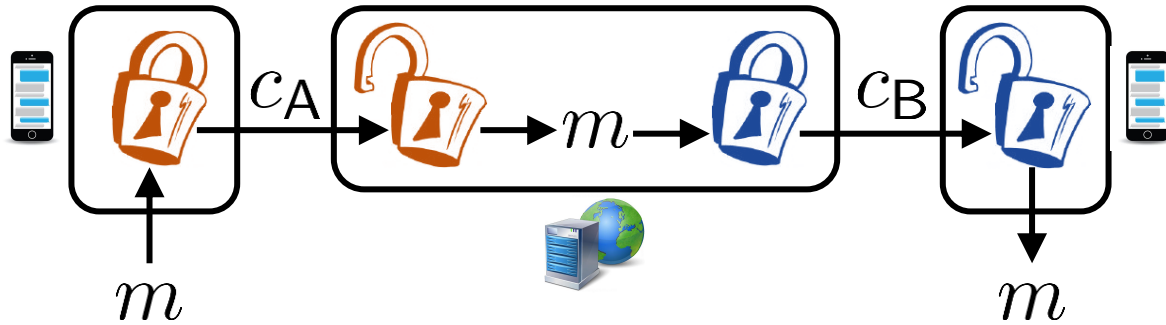
MTPROTO 2.0

MTPROTO 2.0 is **Telegram's** equivalent of the TLS record protocol.



MTPROTO 2.0 is not well-studied.

Cloud Chats



Why not use TLS?

<https://core.telegram.org/techfaq>

Telegram FAQ



Q: Why are you not using X? (insert solution)
While other ways of achieving the same cryptographic goals, undoubtedly, exist, we feel that the present solution is both **robust** and also succeeds at our secondary task of beating unencrypted messengers in terms of **delivery time** and **stability**.

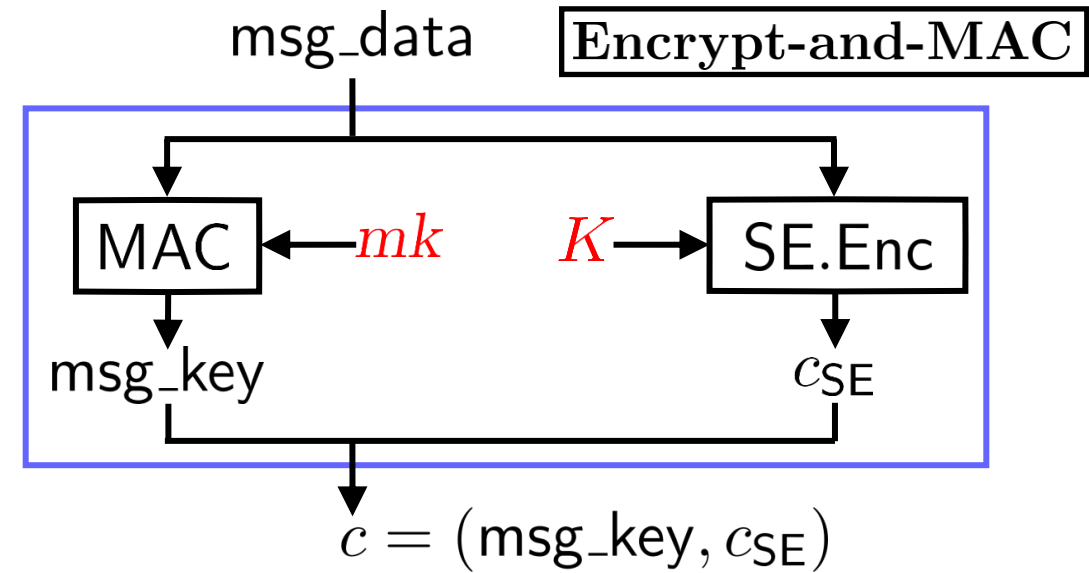
MTPROTO 2.0

MTPROTO 2.0 is **Telegram's** equivalent of the TLS record protocol.

- 2013 ● **Telegram** launched with MTPROTO 1.0.
- 2016 — MTPROTO 1.0 is **not CCA-secure** [JO16].
- 2017 — **Input validation bug** (message replay) [SK17].
Telegram released MTPROTO 2.0.
- 2018 — **Input validation bug** in key exchange [K18].
- 2020 — MTPROTO 2.0 **secure** in symbolic model [MV20].
(assuming ideal building blocks)

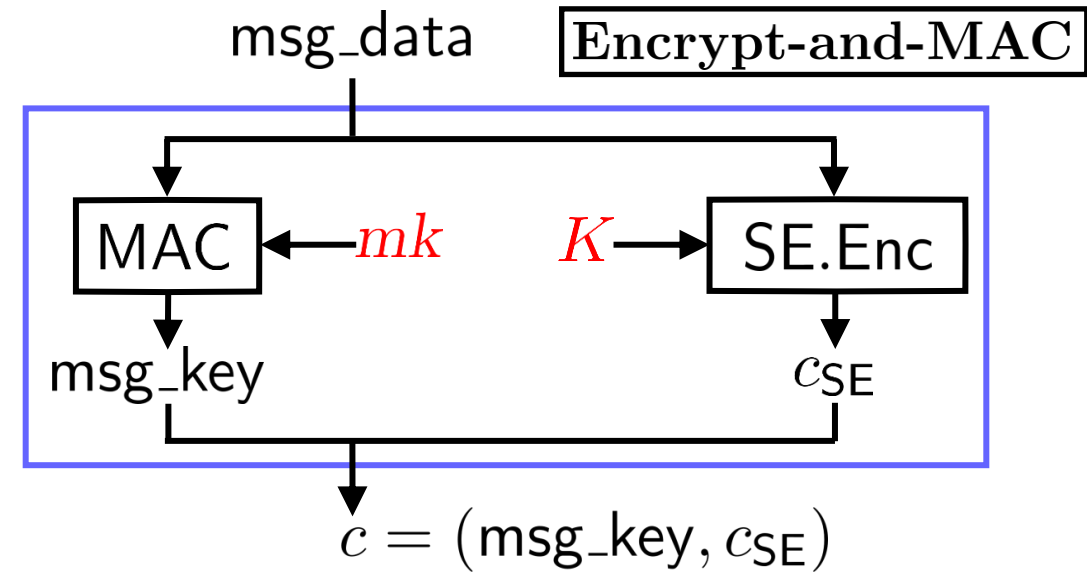
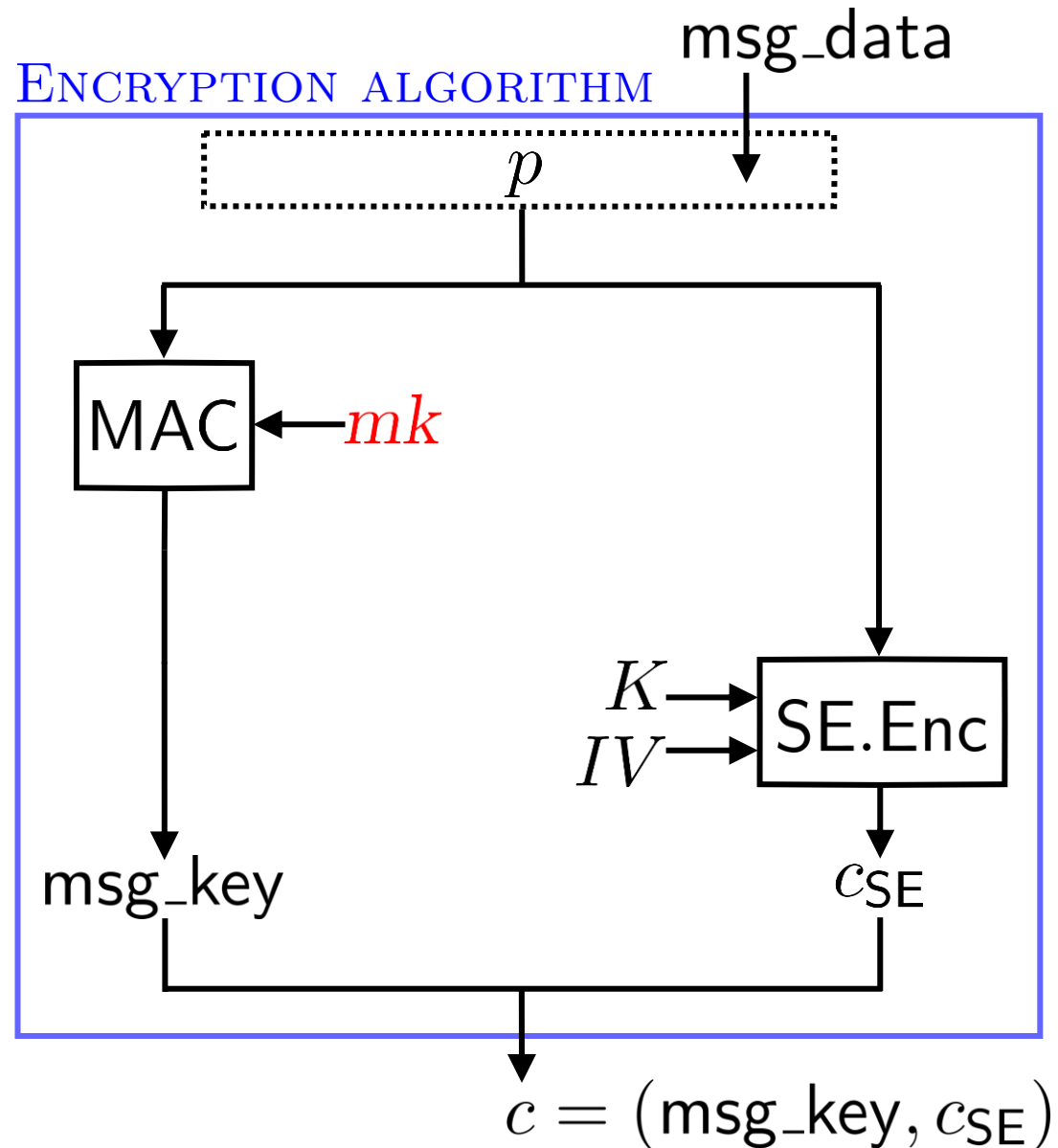
MTPROTO 2.0 is not well-studied.

The Design of MTProto 2.0

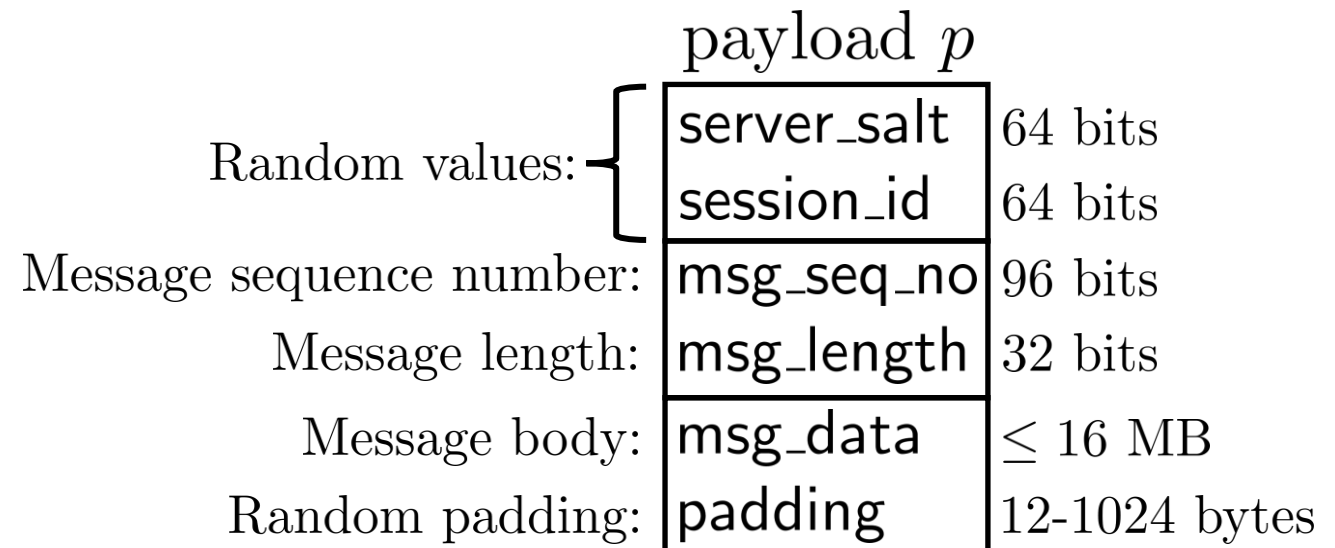


MAC – Message Authentication Code
SE – Symmetric Encryption Scheme

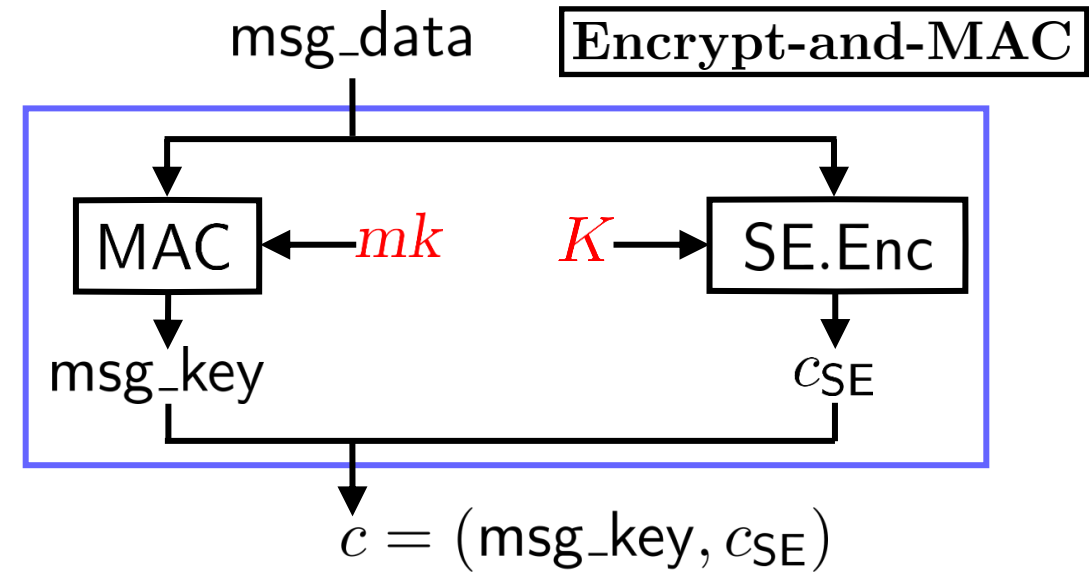
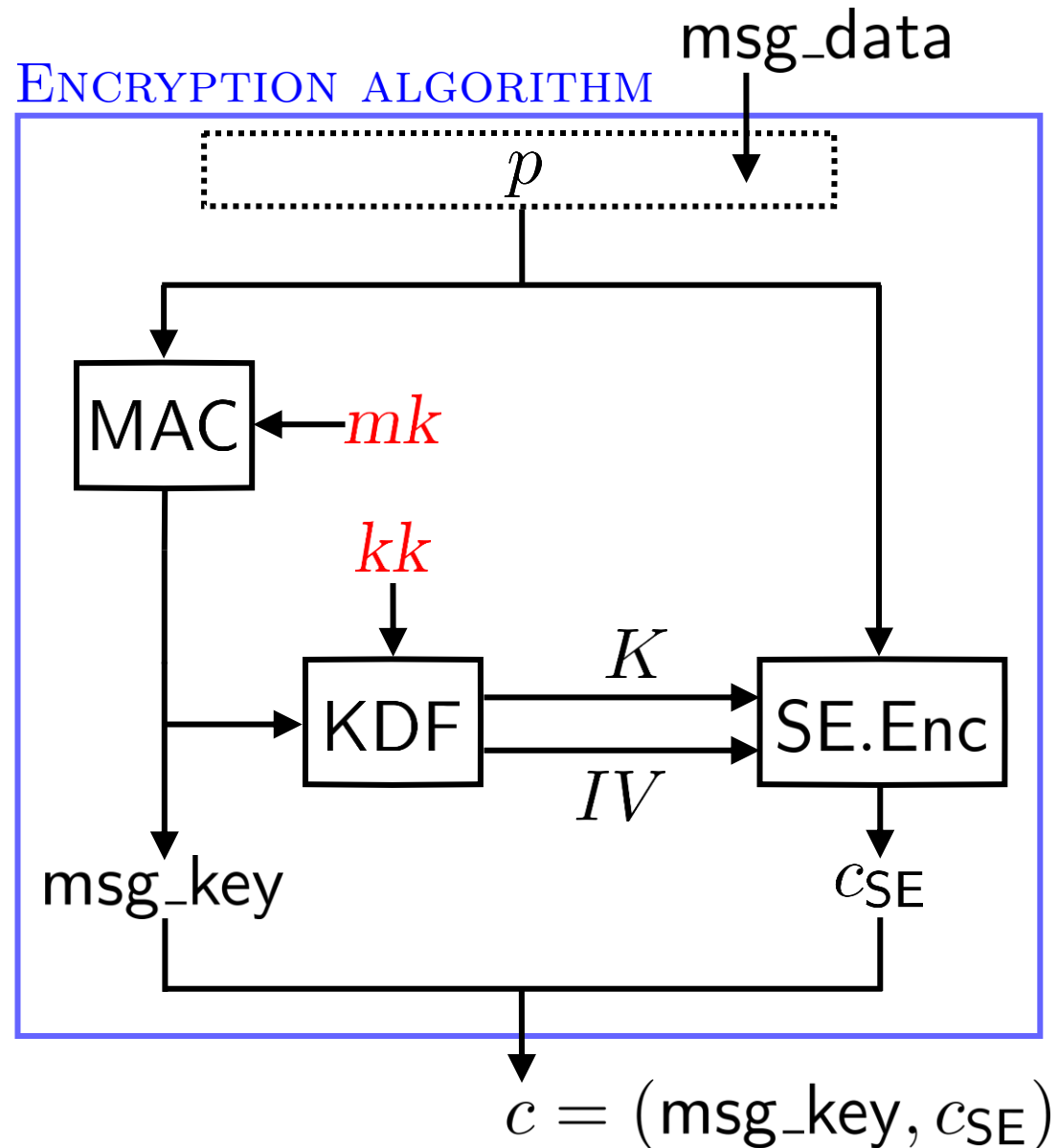
The Design of MTProto 2.0



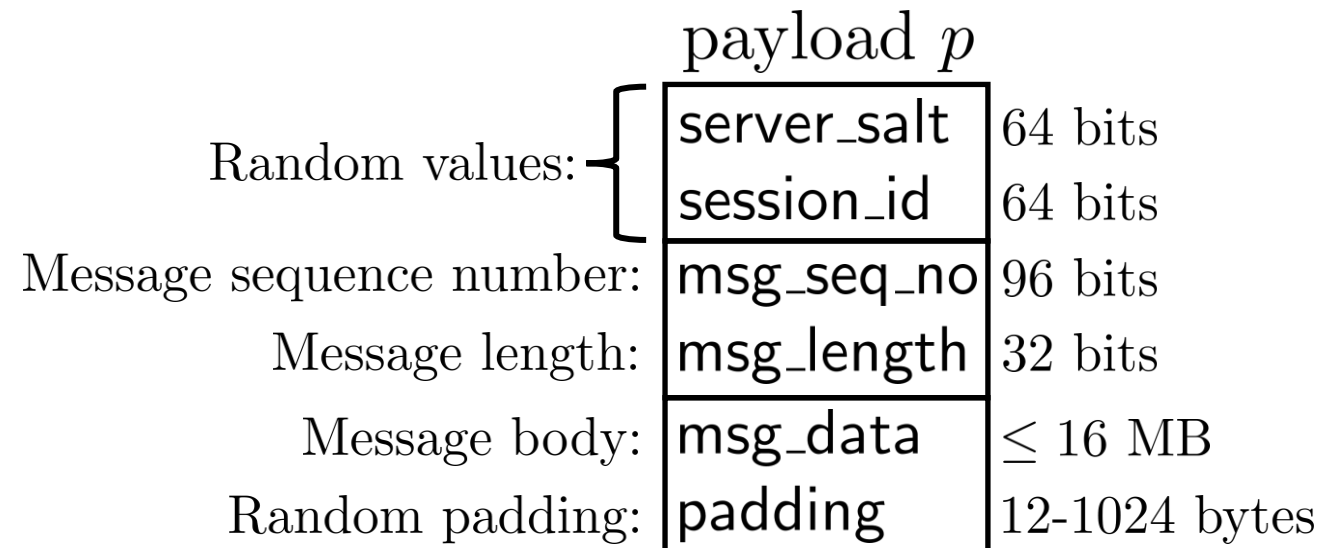
MAC – Message Authentication Code
SE – Symmetric Encryption Scheme



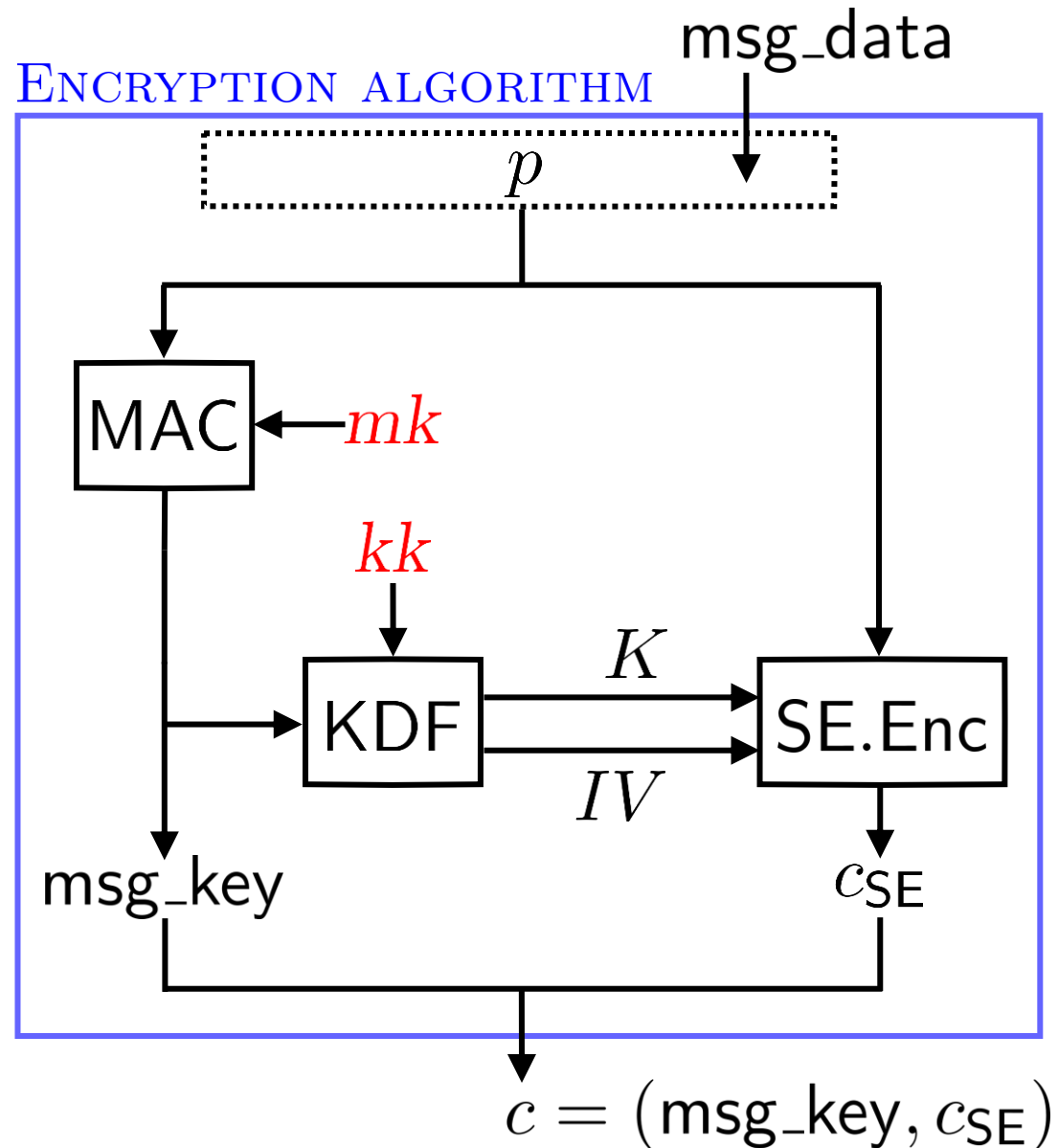
The Design of MTProto 2.0



MAC – Message Authentication Code
SE – Symmetric Encryption Scheme
KDF – Key Derivation Function



The Design of MTPProto 2.0



MTPProto defines **ad-hoc** MAC and KDF schemes.

$\text{MAC}(\textcolor{red}{mk}, p)$

$\text{msg_key} \leftarrow \text{SHA-256}(\textcolor{red}{mk} || p)[64 : 192]$

Return msg_key

$\text{KDF}(\textcolor{red}{kk}, \text{msg_key})$

$(\textcolor{red}{kk}_0, \textcolor{red}{kk}_1) \leftarrow \textcolor{red}{kk}$

$K \leftarrow \text{SHA-256}(\text{msg_key} || \textcolor{red}{kk}_0)$

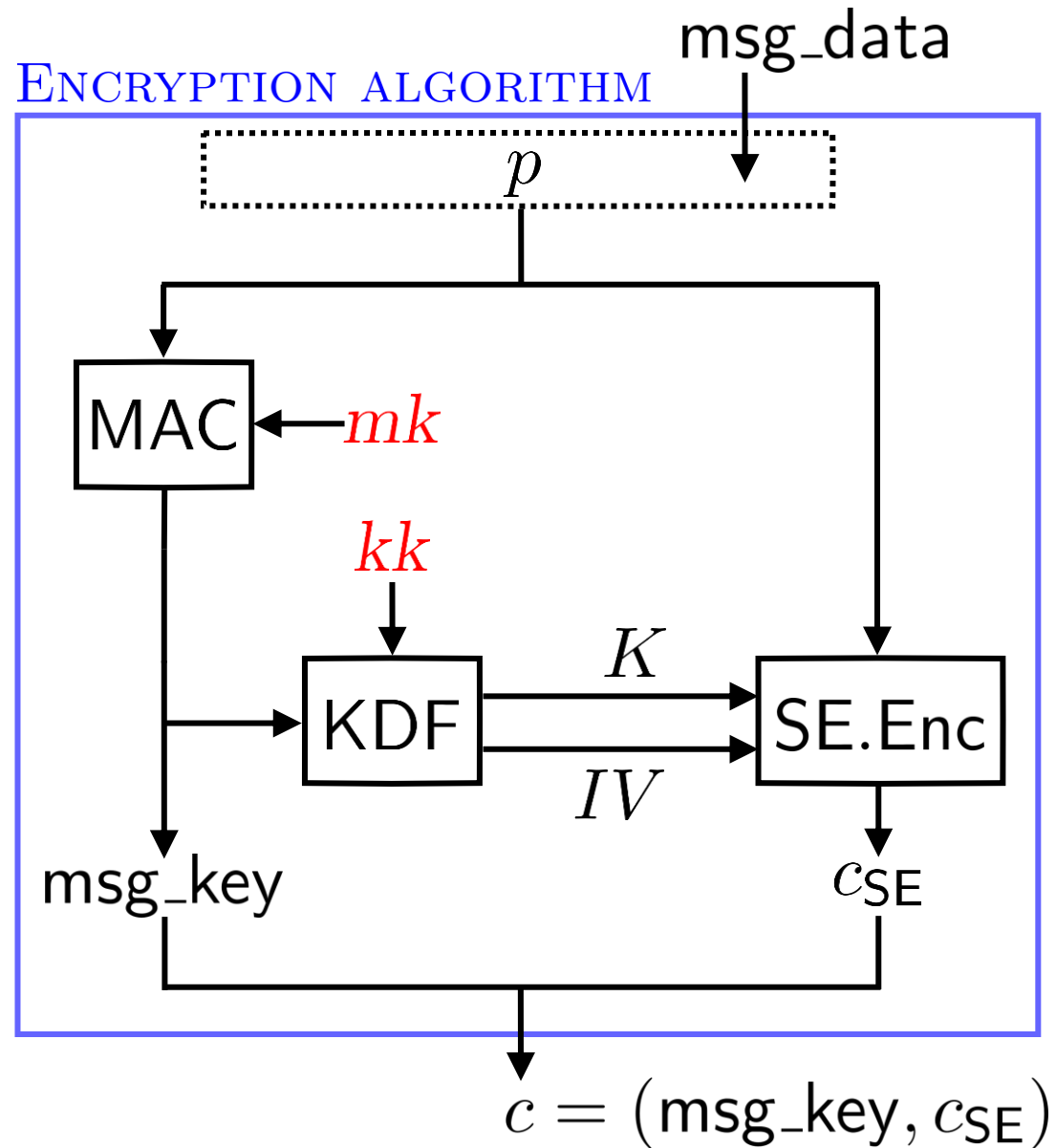
$IV \leftarrow \text{SHA-256}(\textcolor{red}{kk}_1 || \text{msg_key})$

Return K, IV

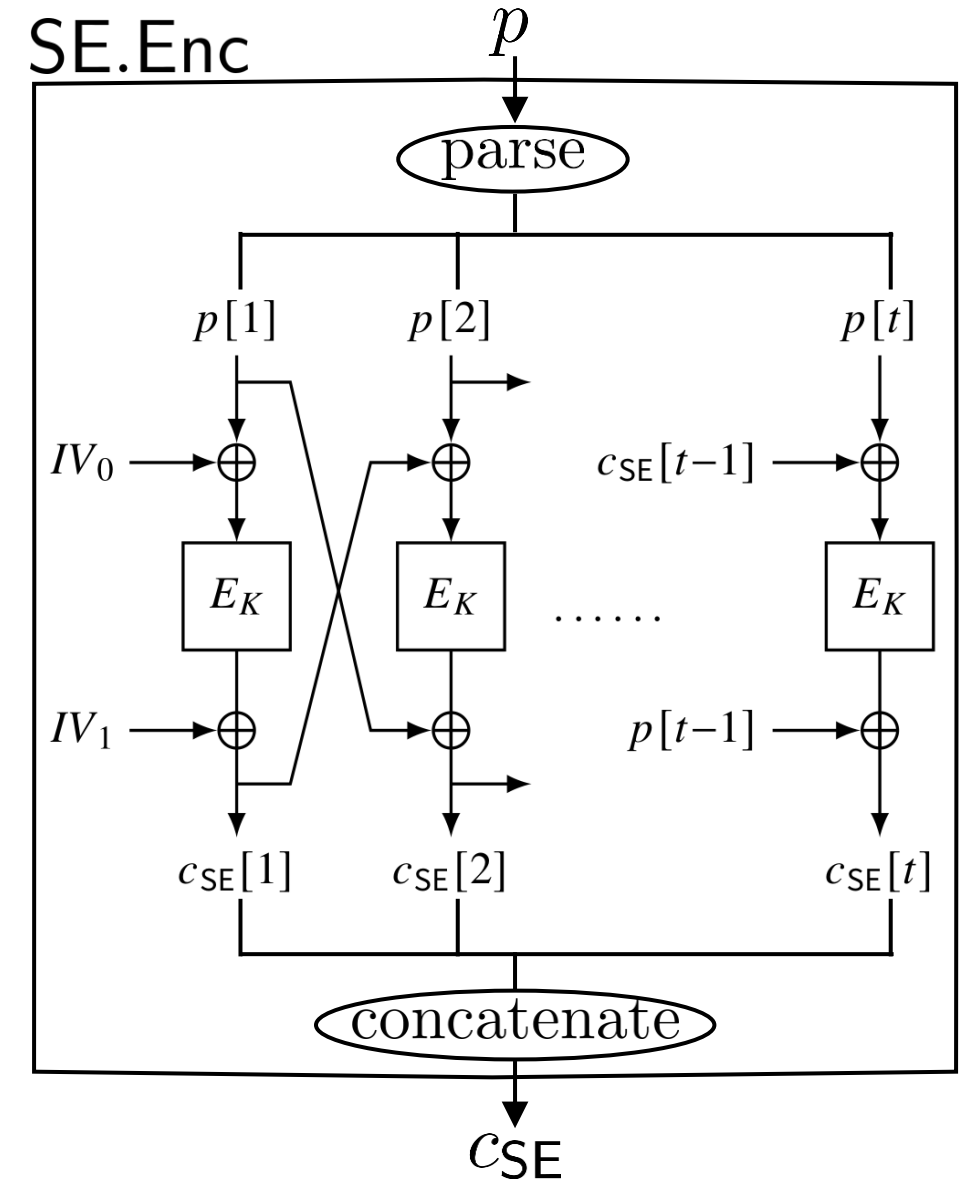
Why invent new MAC
and KDF schemes?



The Design of MTProto 2.0

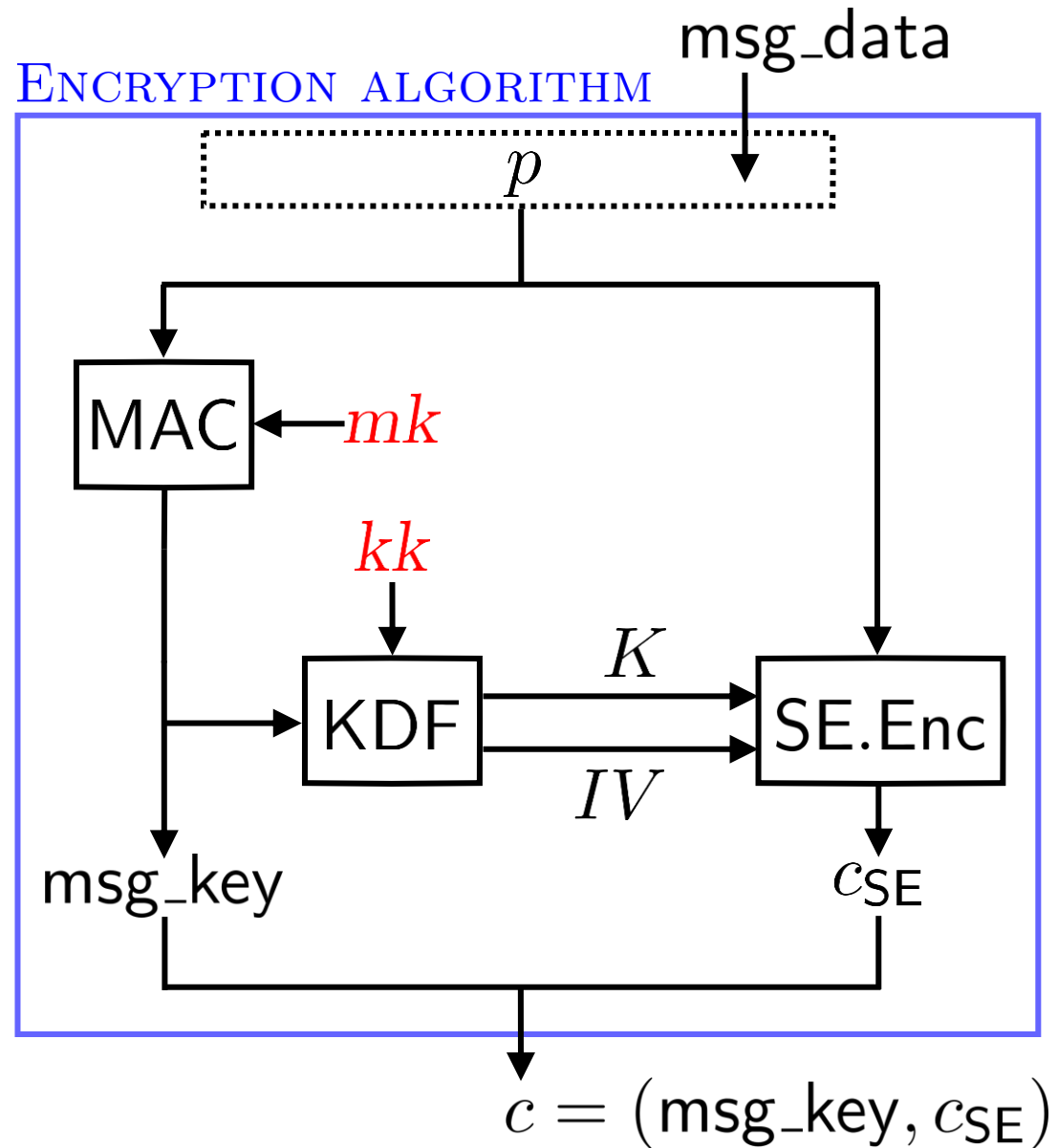


Infinite Garble Extension (IGE) block cipher mode of operations



Not commonly used and not well studied.

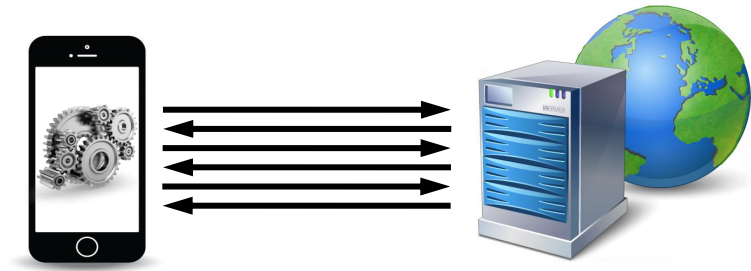
The Design of MTProto 2.0



How are mk and kk derived?

The Design of MTProto 2.0

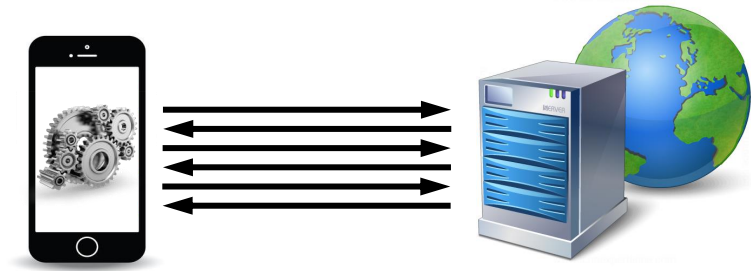
MTProto uses Diffie-Hellman key exchange to agree on a **raw shared secret** g^{xy} .



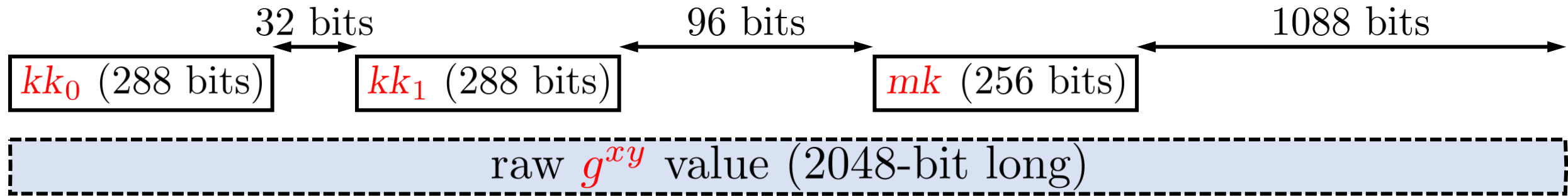
raw g^{xy} value (2048-bit long)

The Design of MTProto 2.0

MTProto uses Diffie-Hellman key exchange to agree on a **raw shared secret** g^{xy} .

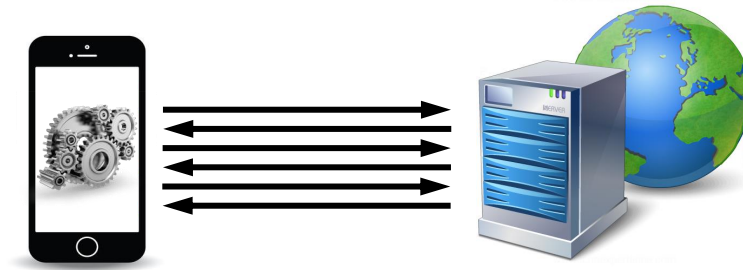


Keys used for **client** \rightarrow **server** encryption.

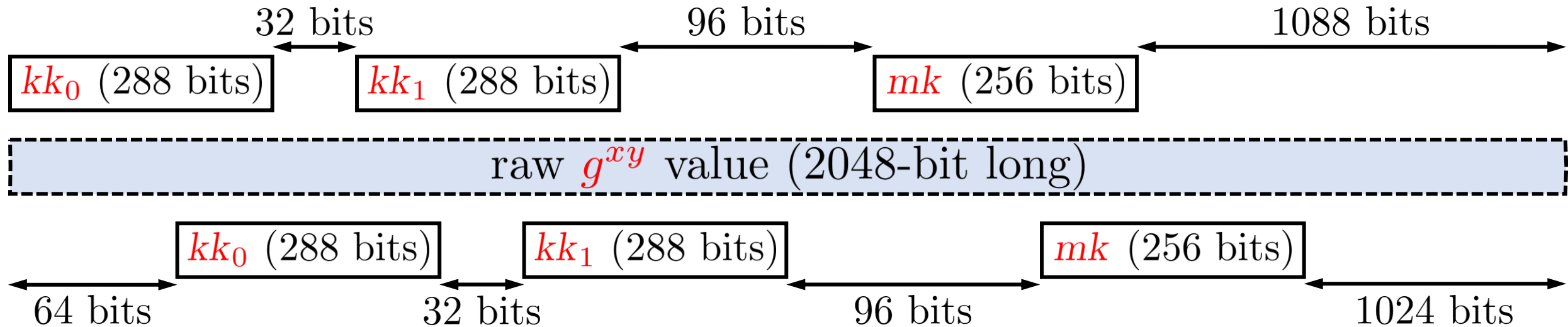


The Design of MTProto 2.0

MTProto uses Diffie-Hellman key exchange to agree on a **raw shared secret** g^{xy} .



Keys used for **client** → **server** encryption.



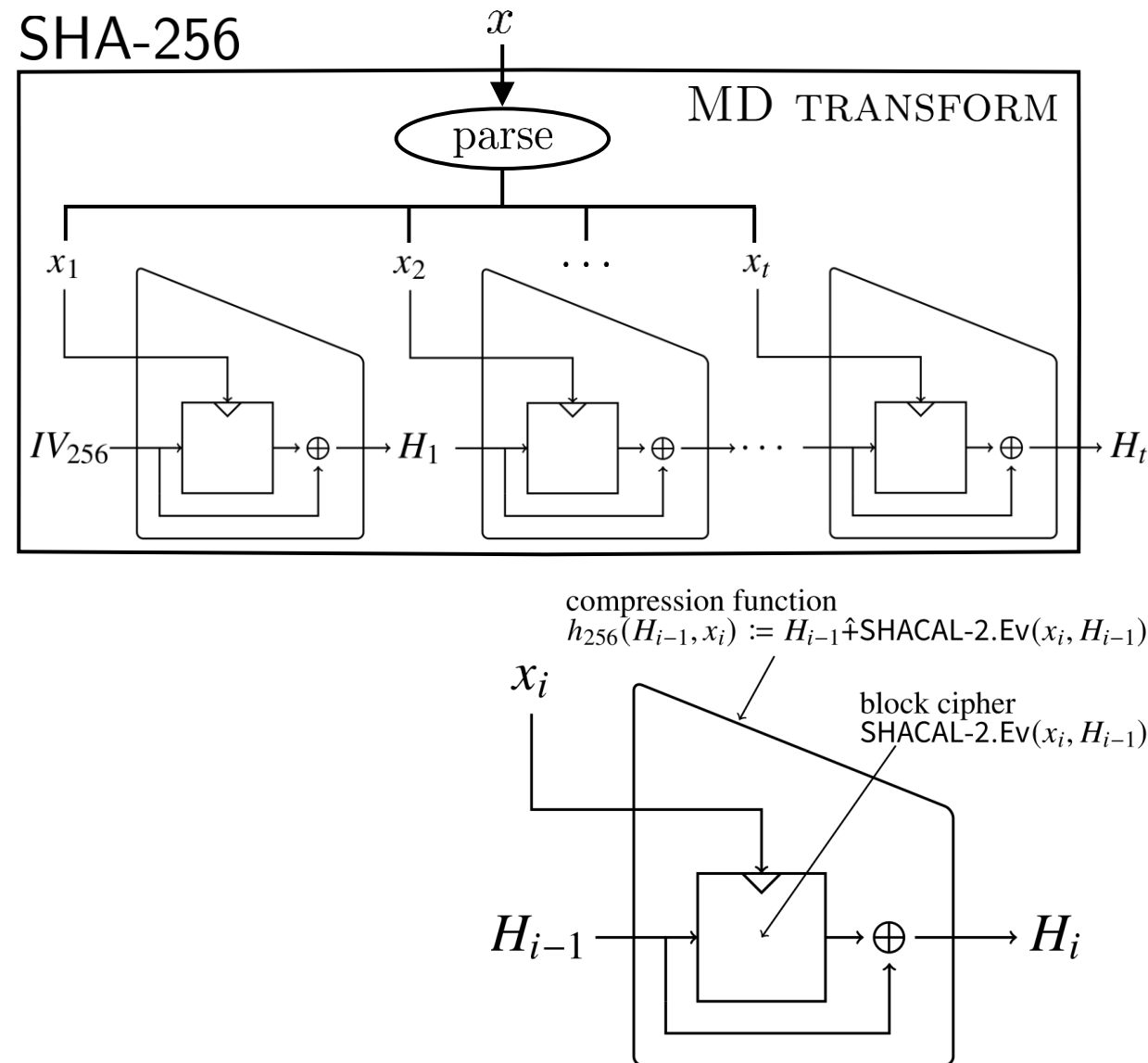
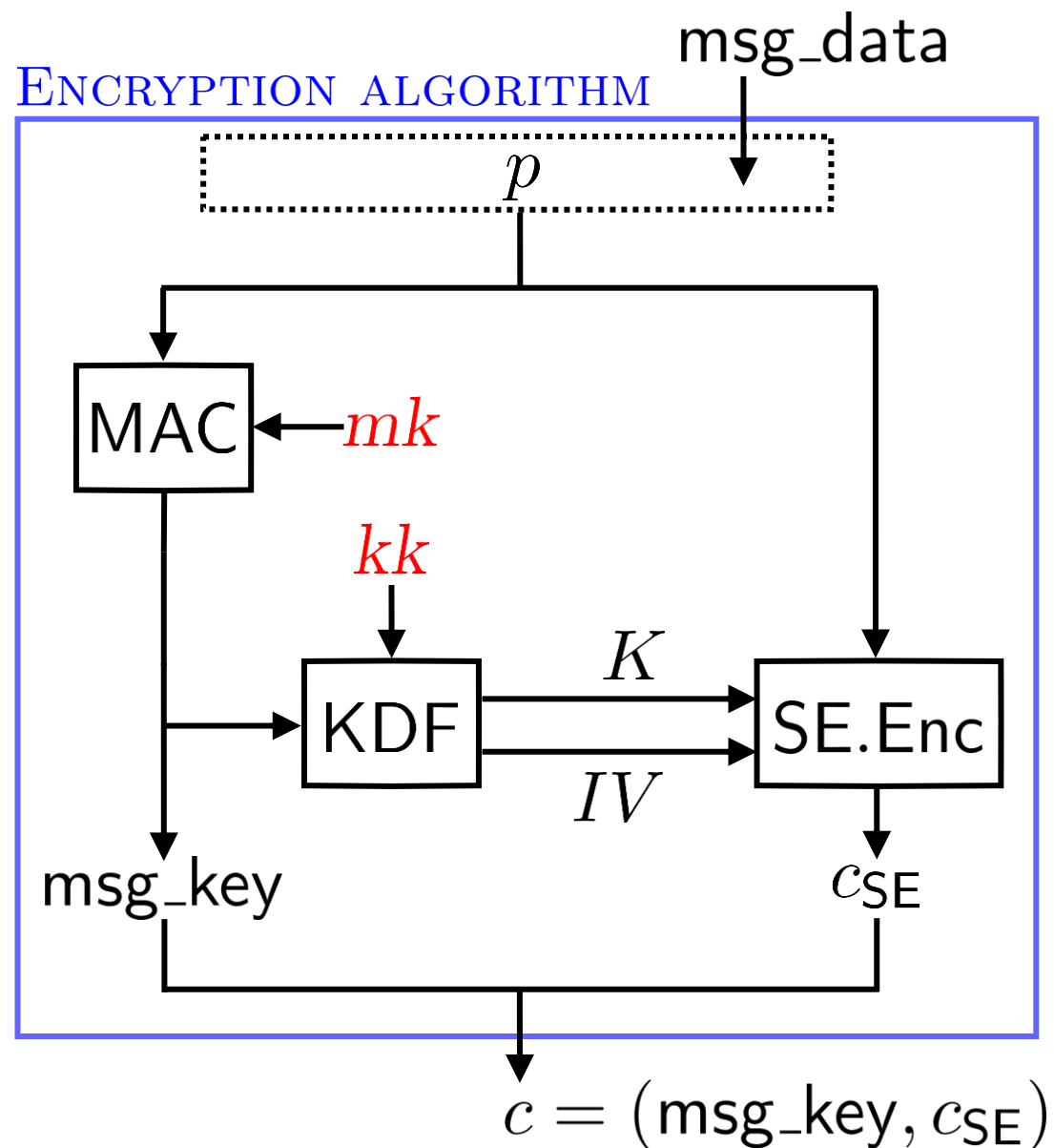
Keys used for **server** → **client** encryption.

Why overlap the key bits?



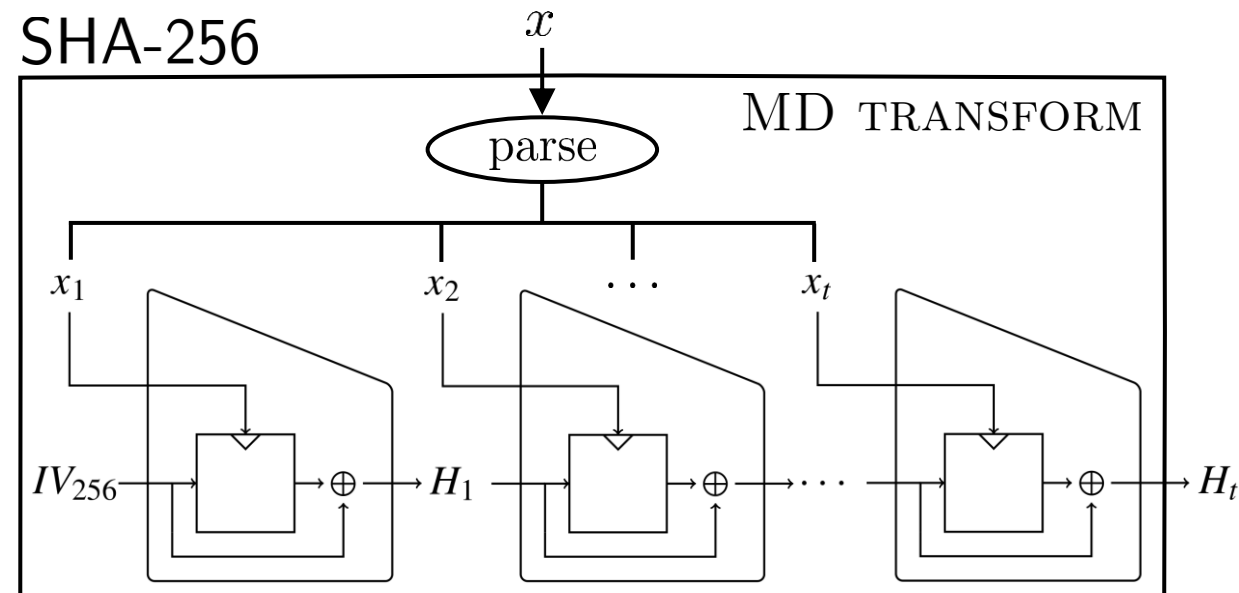
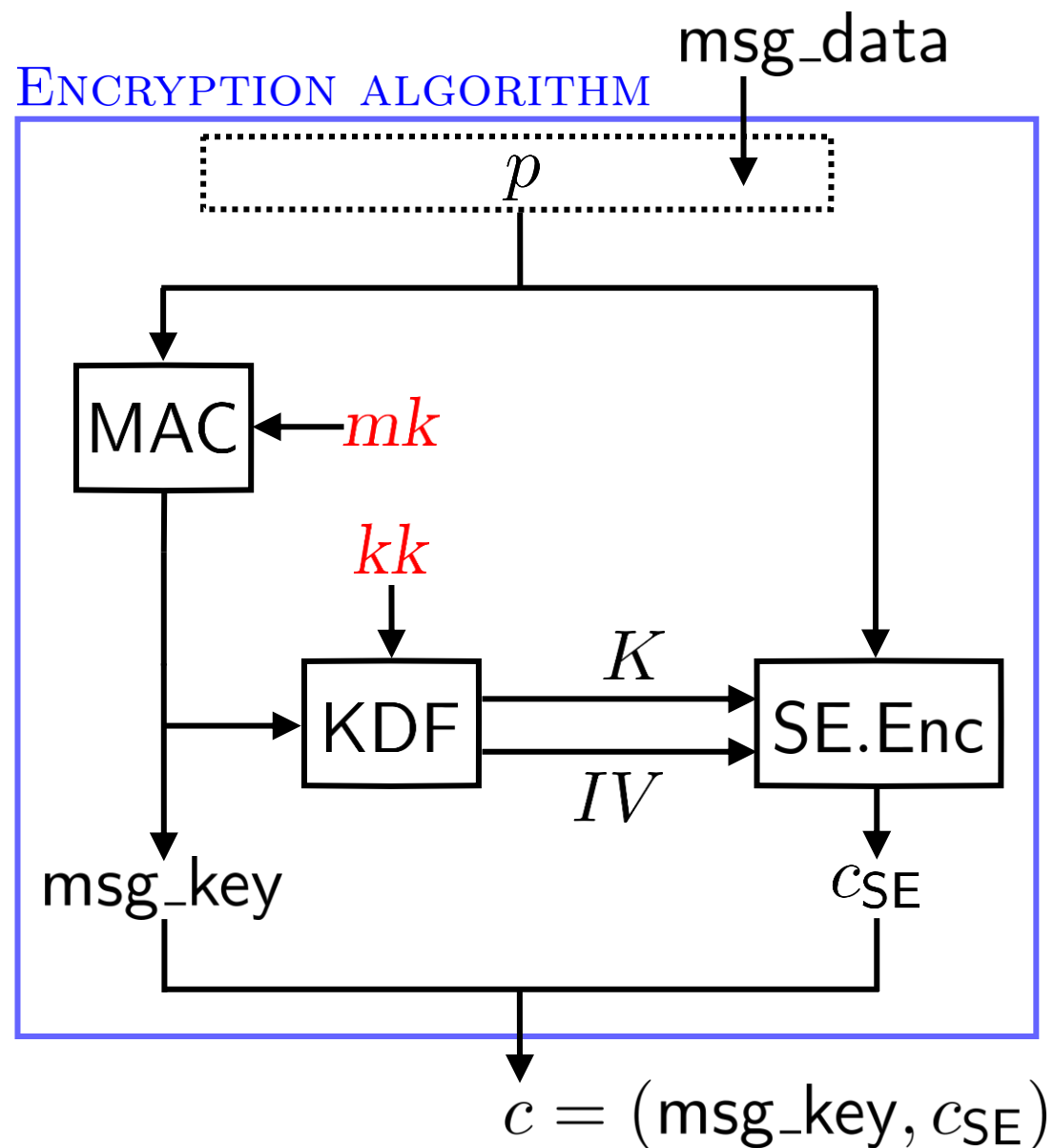
The Design of MTPProto 2.0

We proved a [variant](#) of MTPProto 2.0 is secure.
This comes with [many caveats](#).



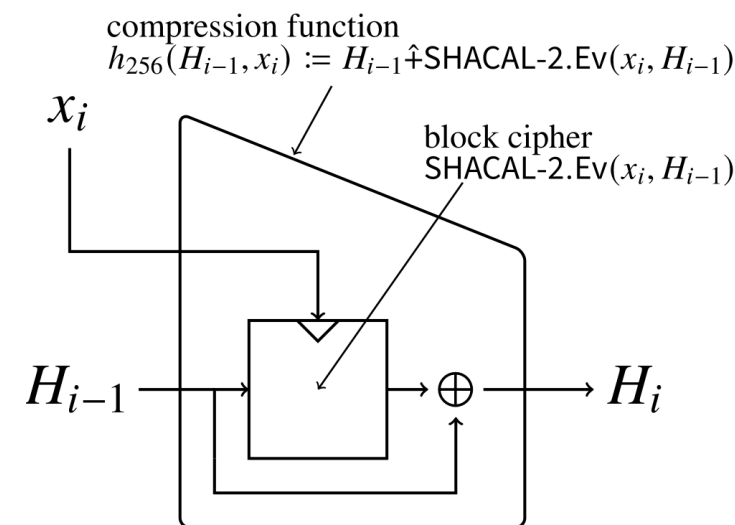
The Design of MTPProto 2.0

We proved a [variant](#) of MTPProto 2.0 is secure.
This comes with [many caveats](#).



We rely on several assumptions about SHACAL-2.

Our novel SHACAL-2 assumptions [need further study](#).



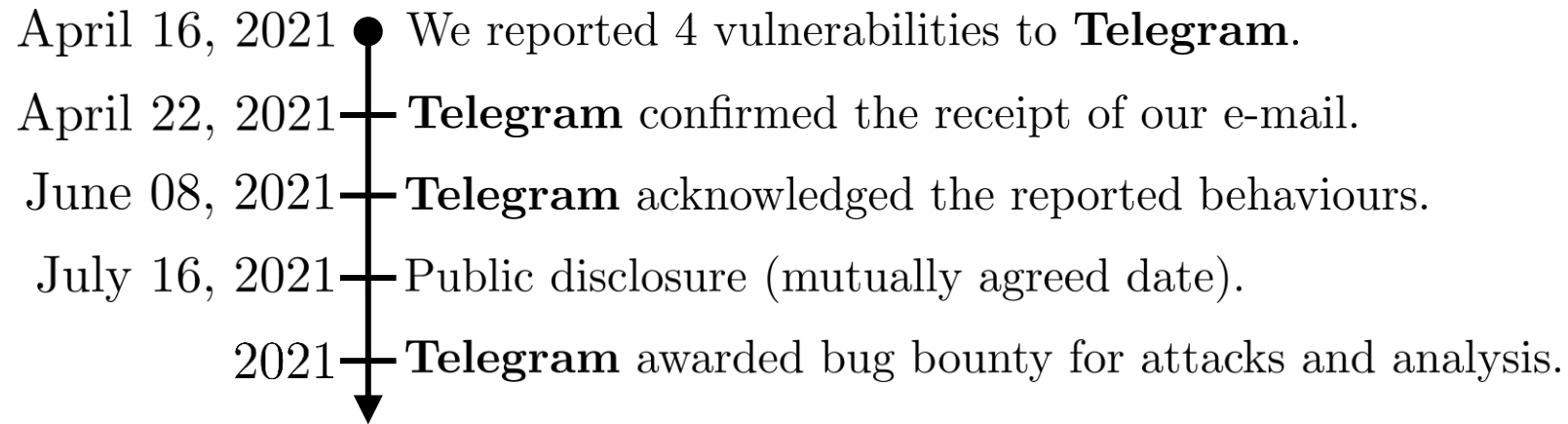
Four Attacks Against Telegram

- April 16, 2021 ● We reported 4 vulnerabilities to **Telegram**.
- April 22, 2021 — **Telegram** confirmed the receipt of our e-mail.
- June 08, 2021 — **Telegram** acknowledged the reported behaviours.
- July 16, 2021 — Public disclosure (mutually agreed date).
- 2021 — **Telegram** awarded bug bounty for attacks and analysis.

All vulnerabilities fixed as of
7.8.1 for **Android**
7.8.3 for **iOS**
2.8.8 for **Desktop**



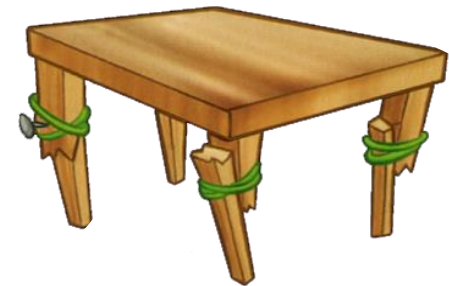
Four Attacks Against Telegram



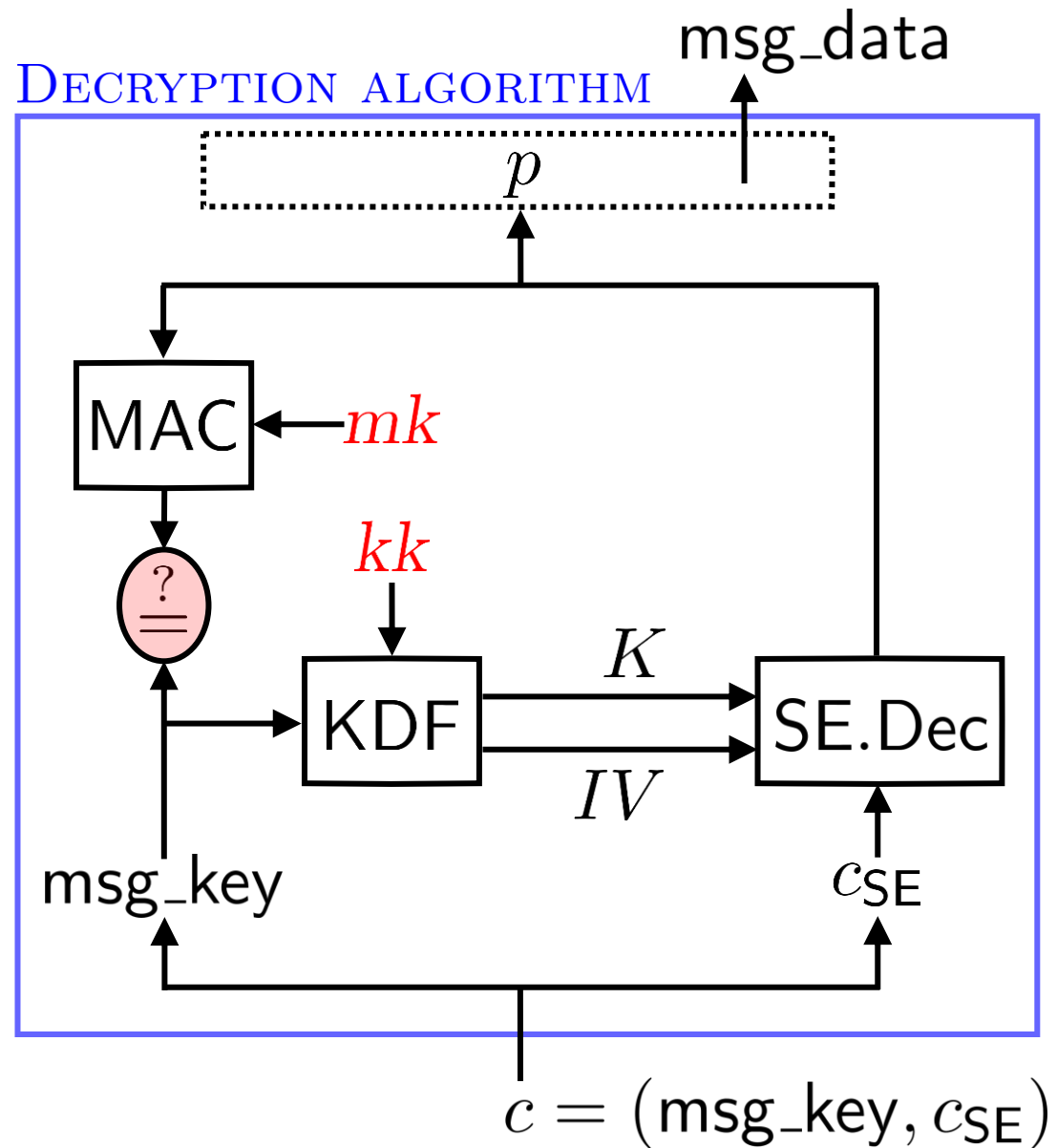
All vulnerabilities fixed as of
7.8.1 for **Android**
7.8.3 for **iOS**
2.8.8 for **Desktop**



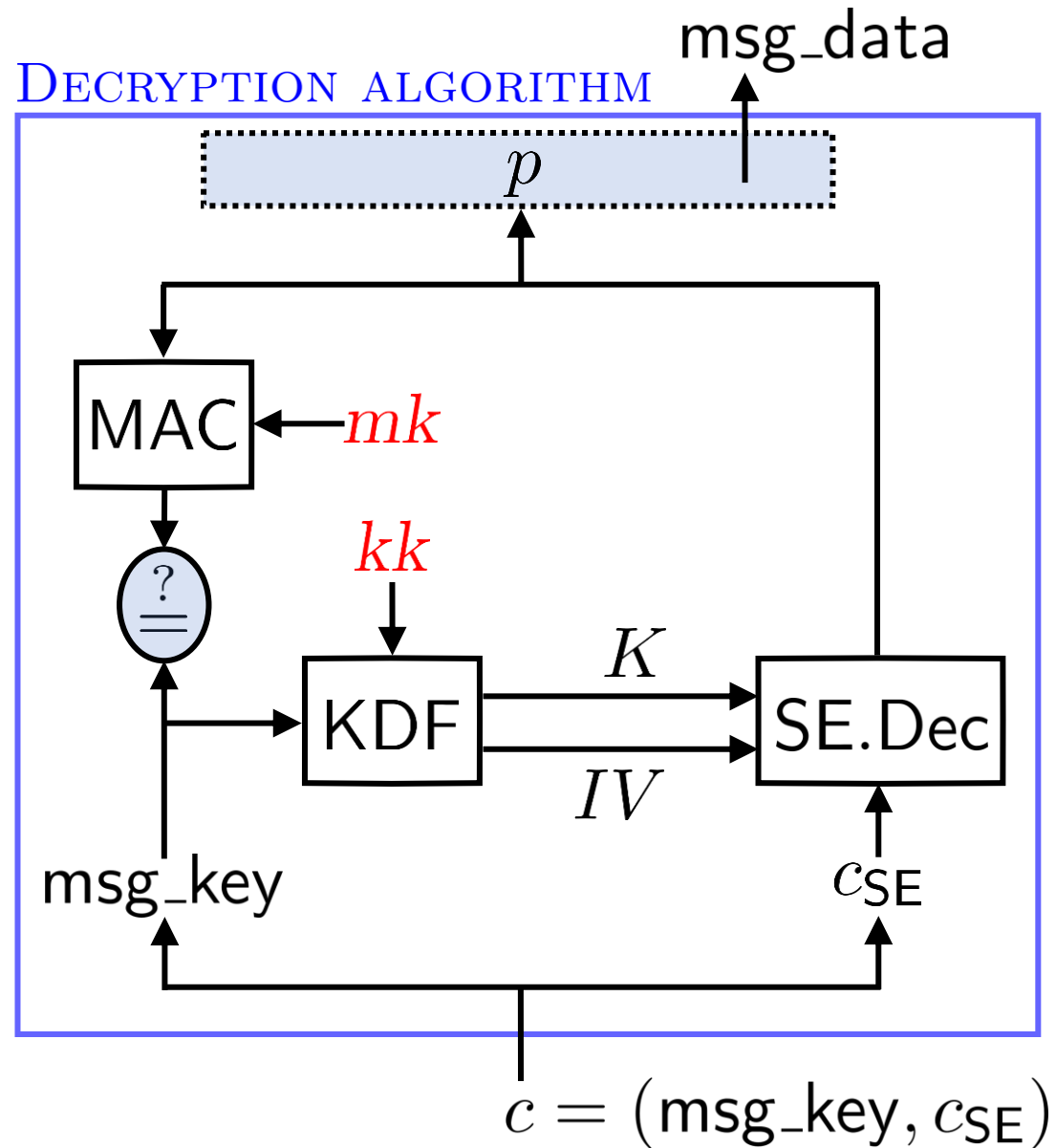
Telegram informed us that they
... do no **security/bugfix releases** except for post-release crash fixes.
(could not commit to **release dates** for specific fixes)
(fixes were rolled out as part of regular updates)
... did not wish to issue **security advisories** at the time of patching.



Timing Side-Channel Attacks



Timing Side-Channel Attacks

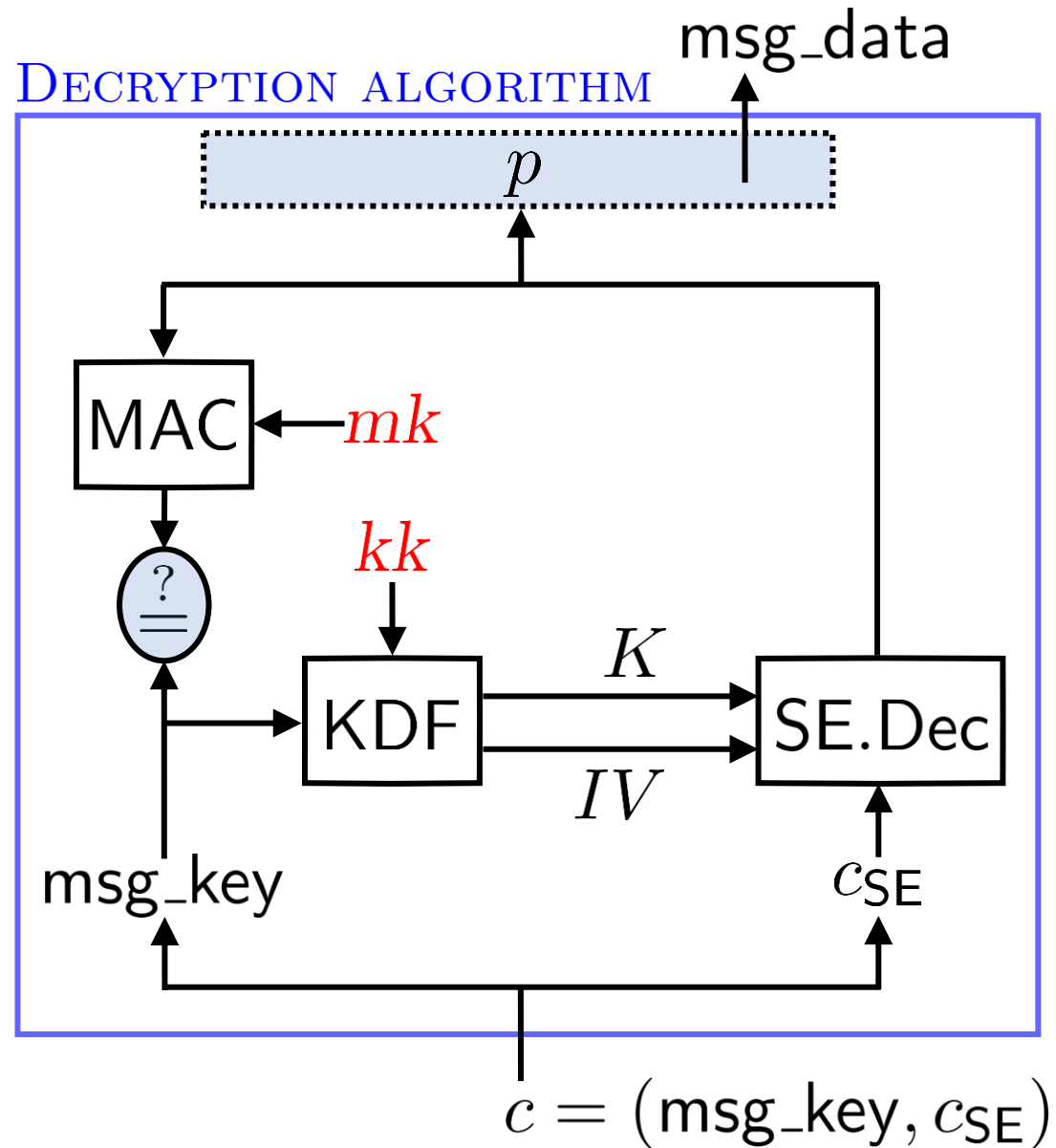


Telegram Desktop

if ($\text{msg_length} > 2^{24}$) then
// MAC verification skipped

payload p	
server_salt	64 bits
session_id	64 bits
msg_seq_no	96 bits
msg_length	32 bits
msg_data	...
padding	...

Timing Side-Channel Attacks



Telegram Desktop

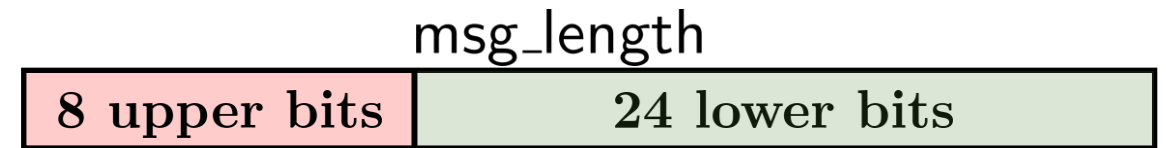
```

if (msg_length > 224) then
    // MAC verification skipped

```

Introduces **3 microsecond** difference.
Remote observer learns up to **8 bits**.

payload p	
server_salt	64 bits
session_id	64 bits
msg_seq_no	96 bits
msg_length	32 bits
msg_data	...
padding	...

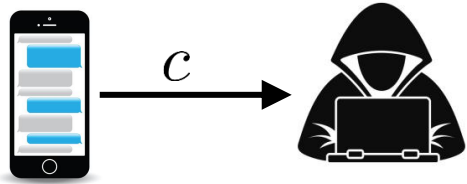


Timing Side-Channel Attacks

We adapt the attack from:

Plaintext Recovery Attacks Against SSH

Martin R. Albrecht, Kenneth G. Paterson and Gaven J. Watson



Telegram Desktop

if ($\text{msg_length} > 2^{24}$) then
// MAC verification skipped

Introduces **3 microsecond** difference.
Remote observer learns up to **8 bits**.

payload p

server_salt	64 bits
session_id	64 bits
msg_seq_no	96 bits
msg_length	32 bits
msg_data	...
padding	...

msg_length

8 upper bits

24 lower bits

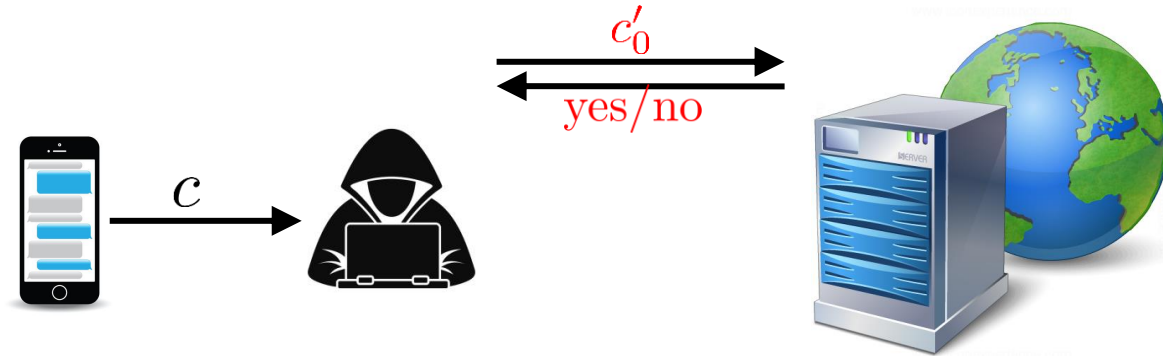
Timing Side-Channel Attacks

We adapt the attack from:

Plaintext Recovery Attacks Against SSH

Martin R. Albrecht, Kenneth G. Paterson and Gaven J. Watson

Build new ciphertexts c'_i by reshuffling c as follows:
move **different blocks of c** into the **2nd block of c'_i** .

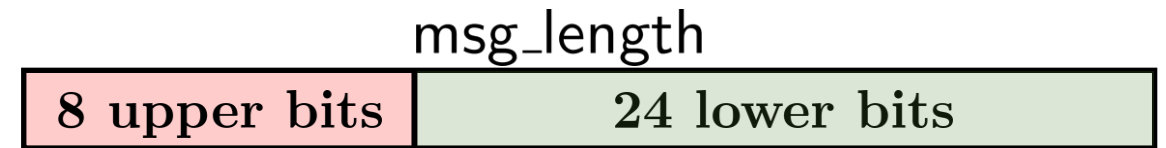


Telegram Desktop

if ($\text{msg_length} > 2^{24}$) then
// MAC verification skipped

Introduces **3 microsecond** difference.
Remote observer learns up to **8 bits**.

payload p	
server_salt	64 bits
session_id	64 bits
msg_seq_no	96 bits
msg_length	32 bits
msg_data	...
padding	...



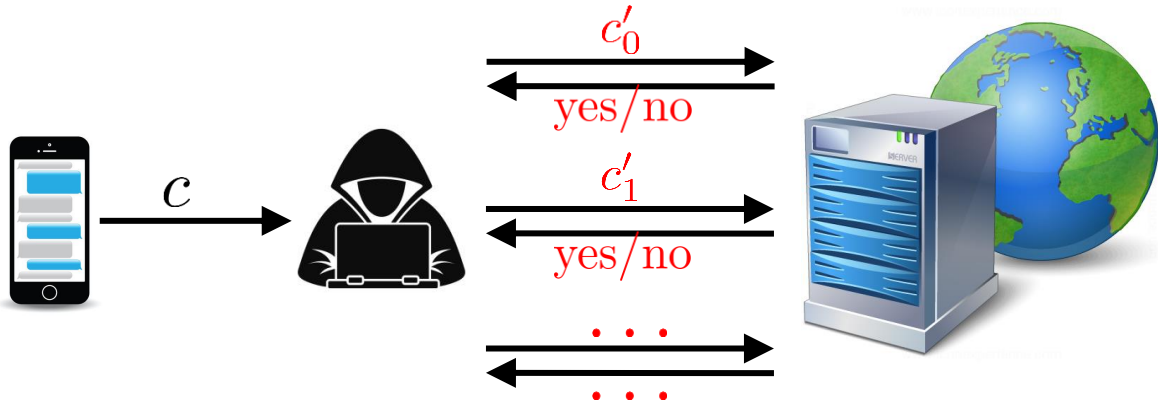
Timing Side-Channel Attacks

We adapt the attack from:

Plaintext Recovery Attacks Against SSH

Martin R. Albrecht, Kenneth G. Paterson and Gaven J. Watson

Build new ciphertexts c'_i by reshuffling c as follows:
move **different blocks of c** into the **2nd block of c'_i** .



“Are the upper 8 bits of msg_length (encrypted in c'_i) equal to 00000000?”

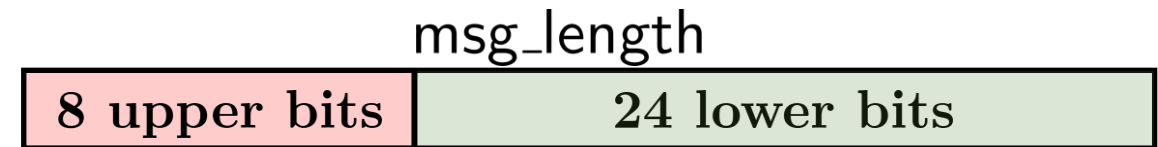
We **recover** (up to) **8 bits** of msg_data per query!

Telegram Desktop

if ($msg_length > 2^{24}$) then
// MAC verification skipped

Introduces **3 microsecond** difference.
Remote observer learns up to **8 bits**.

payload p	
server_salt	64 bits
session_id	64 bits
msg_seq_no	96 bits
msg_length	32 bits
msg_data	...
padding	...



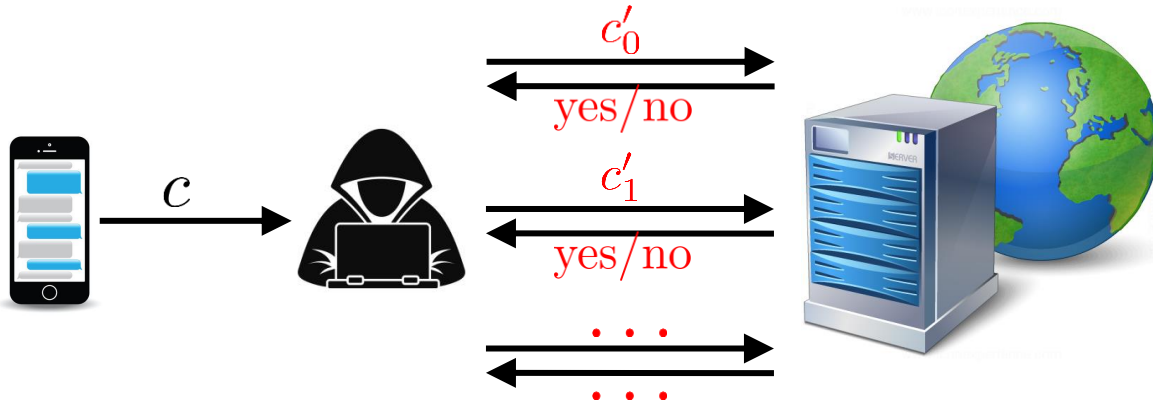
Timing Side-Channel Attacks

We adapt the attack from:

Plaintext Recovery Attacks Against SSH

Martin R. Albrecht, Kenneth G. Paterson and Gaven J. Watson

Build new ciphertexts c'_i by reshuffling c as follows:
move **different blocks of c** into the **2nd block of c'_i** .



“Are the upper 8 bits of msg_length (encrypted in c'_i) equal to 00000000?”

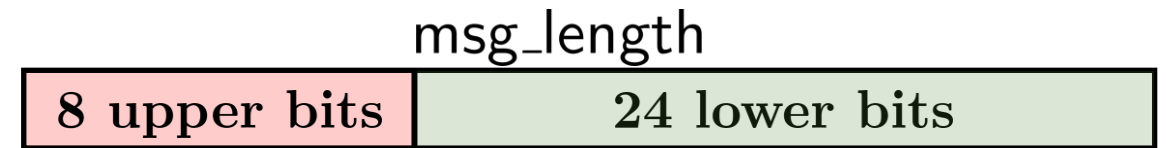
We **recover** (up to) 8 bits of msg_data per query!

Telegram Desktop

if ($\text{msg_length} > 2^{24}$) then
// MAC verification skipped

Introduces **3 microsecond** difference.
Remote observer learns up to **8 bits**.

payload p	
server_salt	64 bits
session_id	64 bits
msg_seq_no	96 bits
msg_length	32 bits
msg_data	...
padding	...



Three **official clients** checked p before MAC.

Telegram Desktop

Telegram Android

Telegram iOS

Each client did it in a different way.

Each client presented a timing side-channel.

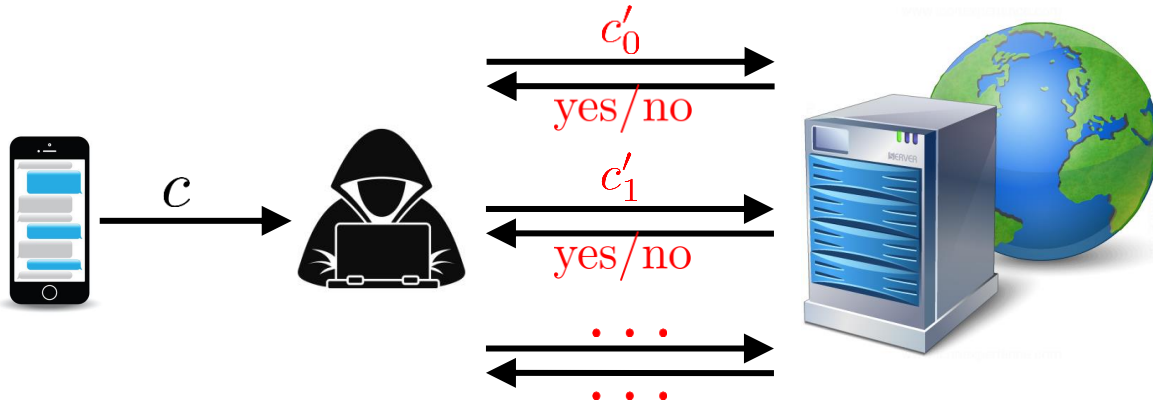
Timing Side-Channel Attacks

We adapt the attack from:

Plaintext Recovery Attacks Against SSH

Martin R. Albrecht, Kenneth G. Paterson and Gaven J. Watson

Build new ciphertexts c'_i by reshuffling c as follows:
move **different blocks of c** into the **2nd block of c'_i** .



“Are the upper 8 bits of msg_length (encrypted in c'_i) equal to 00000000?”

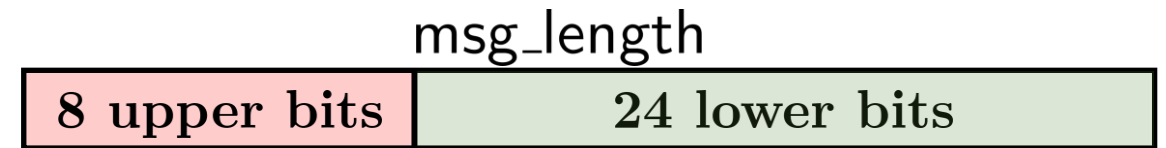
We **recover** (up to) 8 bits of msg_data per query!

Telegram Desktop

if (msg_length > 2^{24}) then
// MAC verification skipped

Introduces **3 microsecond** difference.
Remote observer learns up to **8 bits**.

payload p	
server_salt	64 bits
session_id	64 bits
msg_seq_no	96 bits
msg_length	32 bits
msg_data	...
padding	...



Three **official clients** checked p before MAC.

Telegram Desktop
Telegram Android
Telegram iOS

Each client did it in a different way.

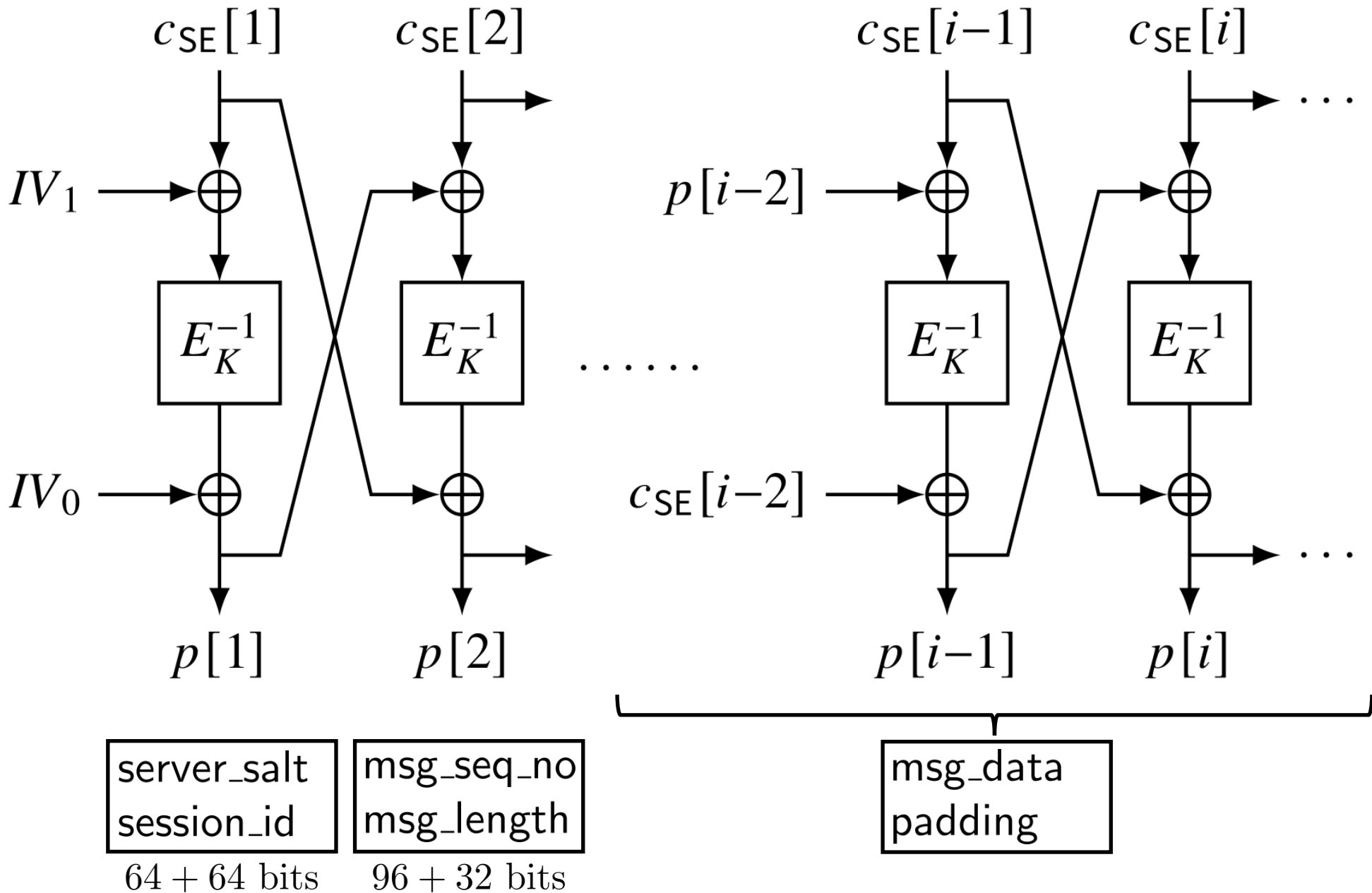
Each client presented a timing side-channel.

This highlights a **brittle design**.

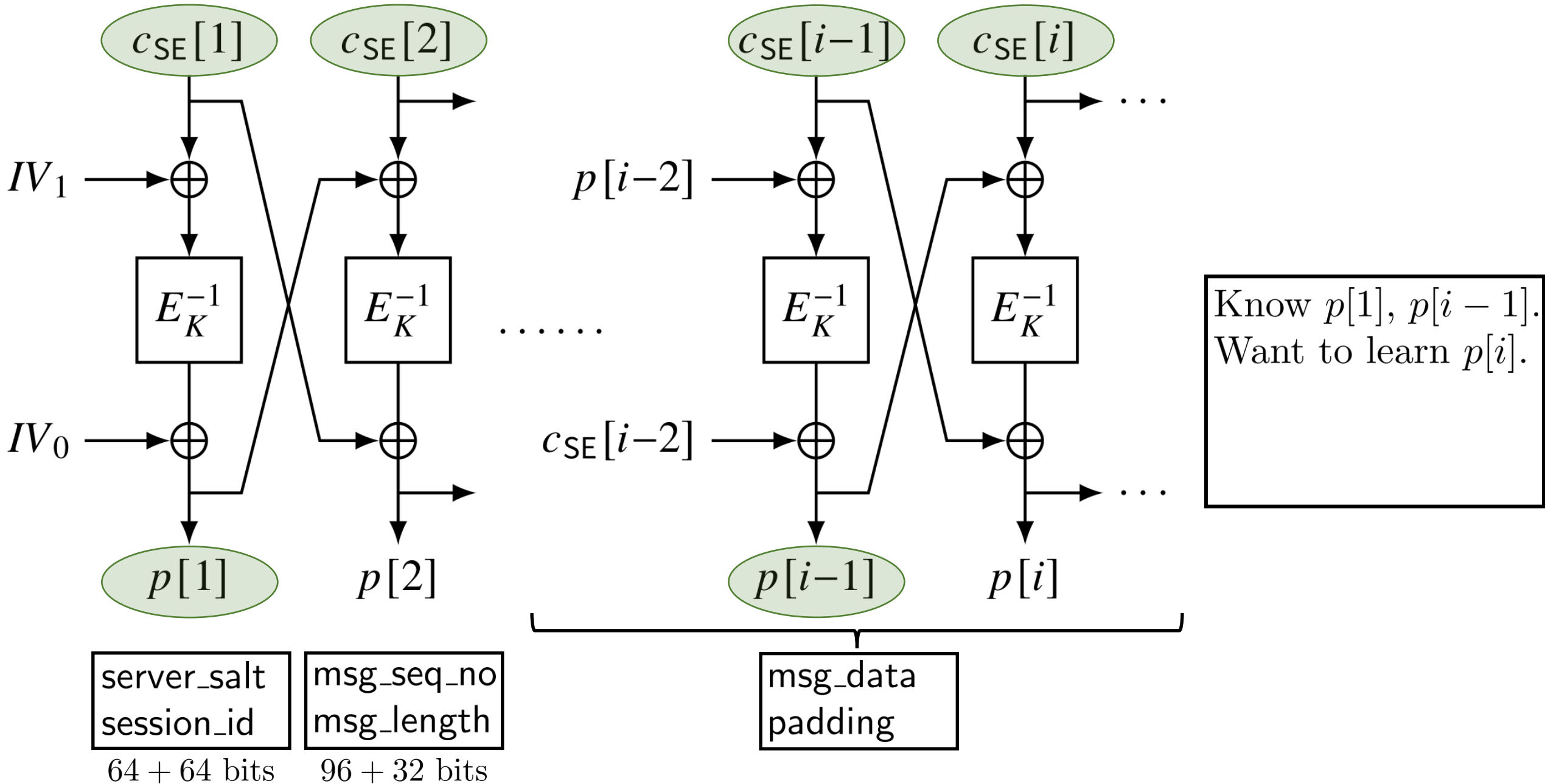
Encrypt-and-MAC requires to **decrypt untrusted data**.

Would be **safer to protect integrity of ciphertext**.

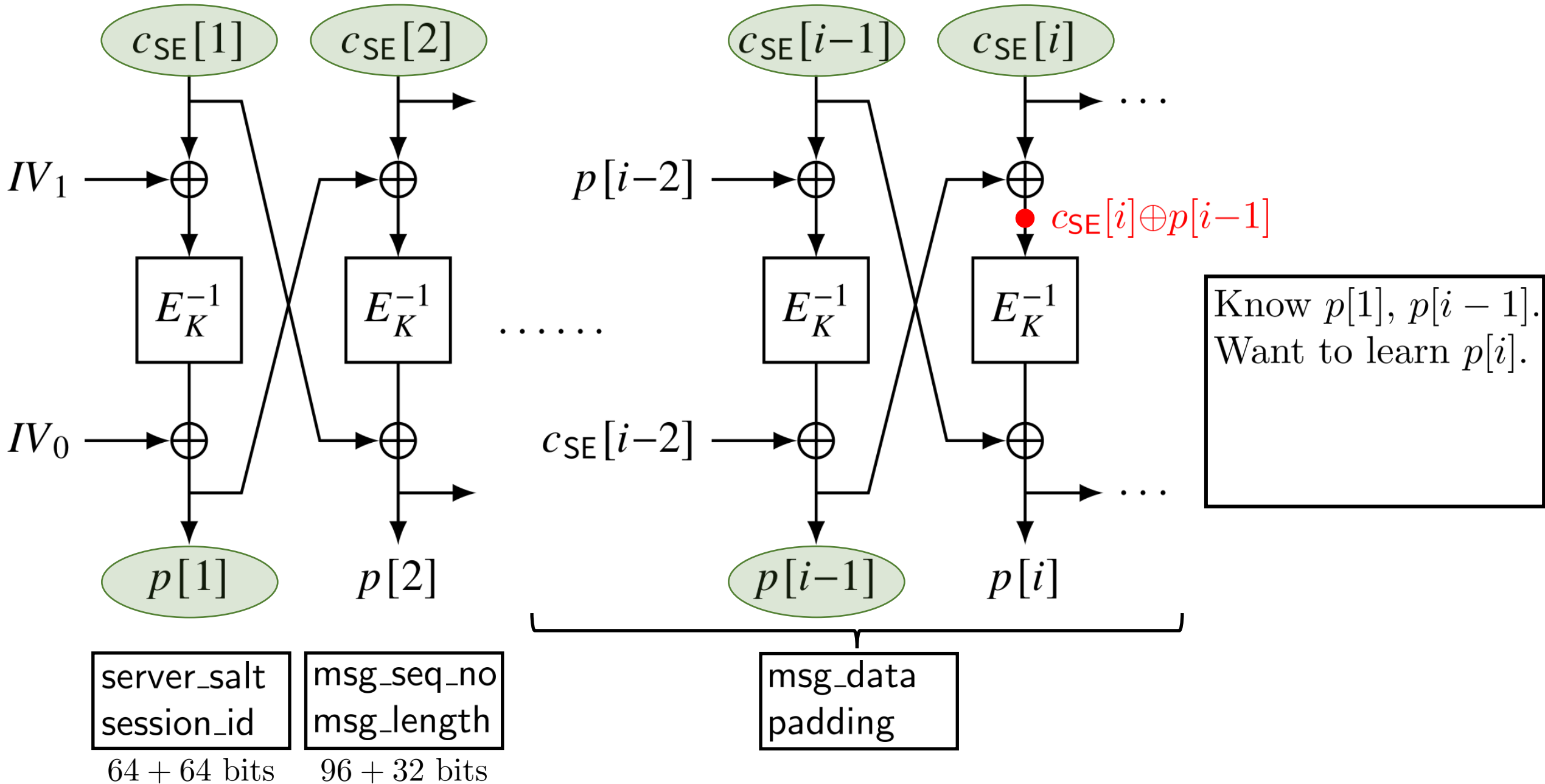
Timing Side-Channel Attacks



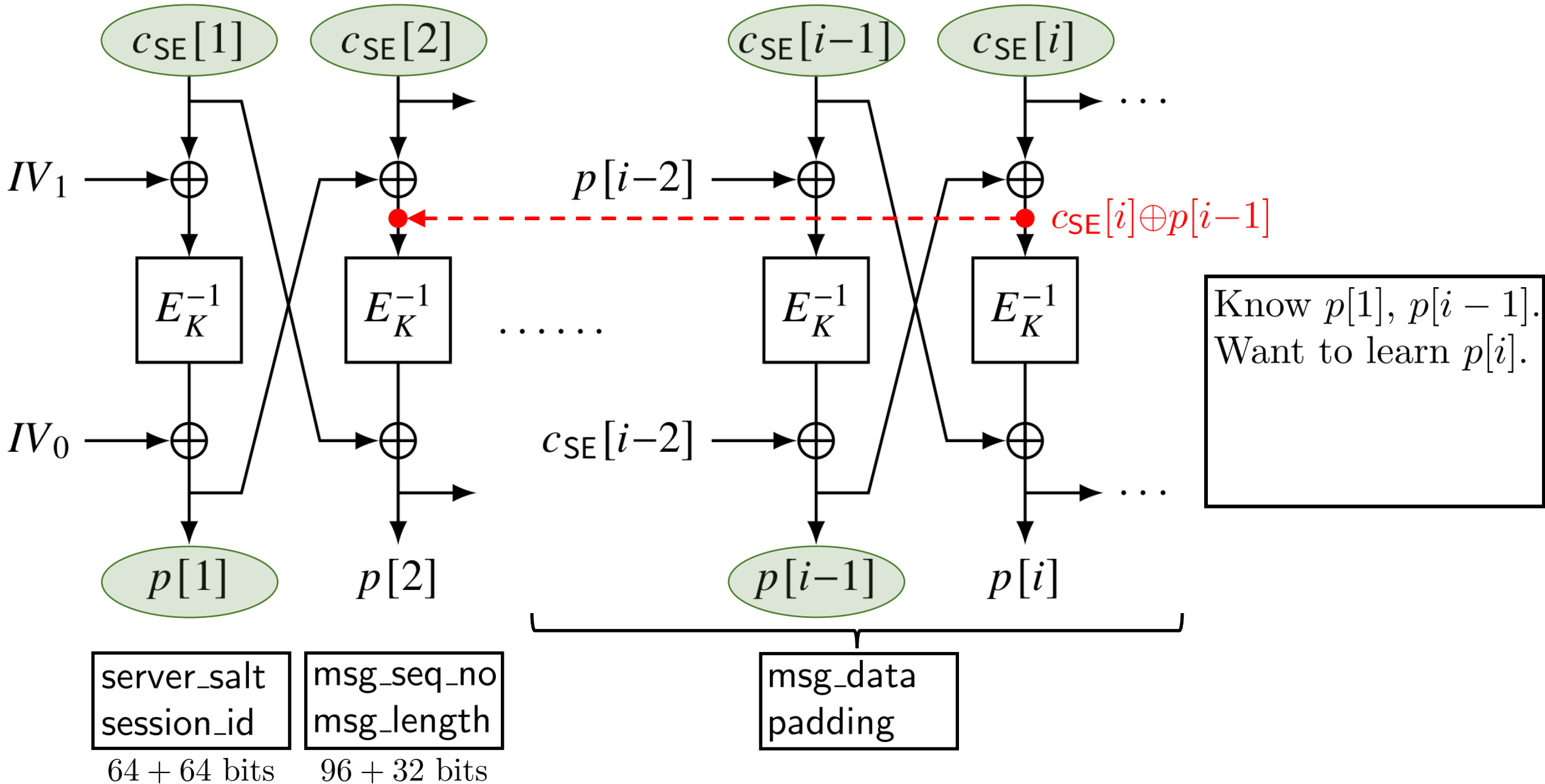
Timing Side-Channel Attacks



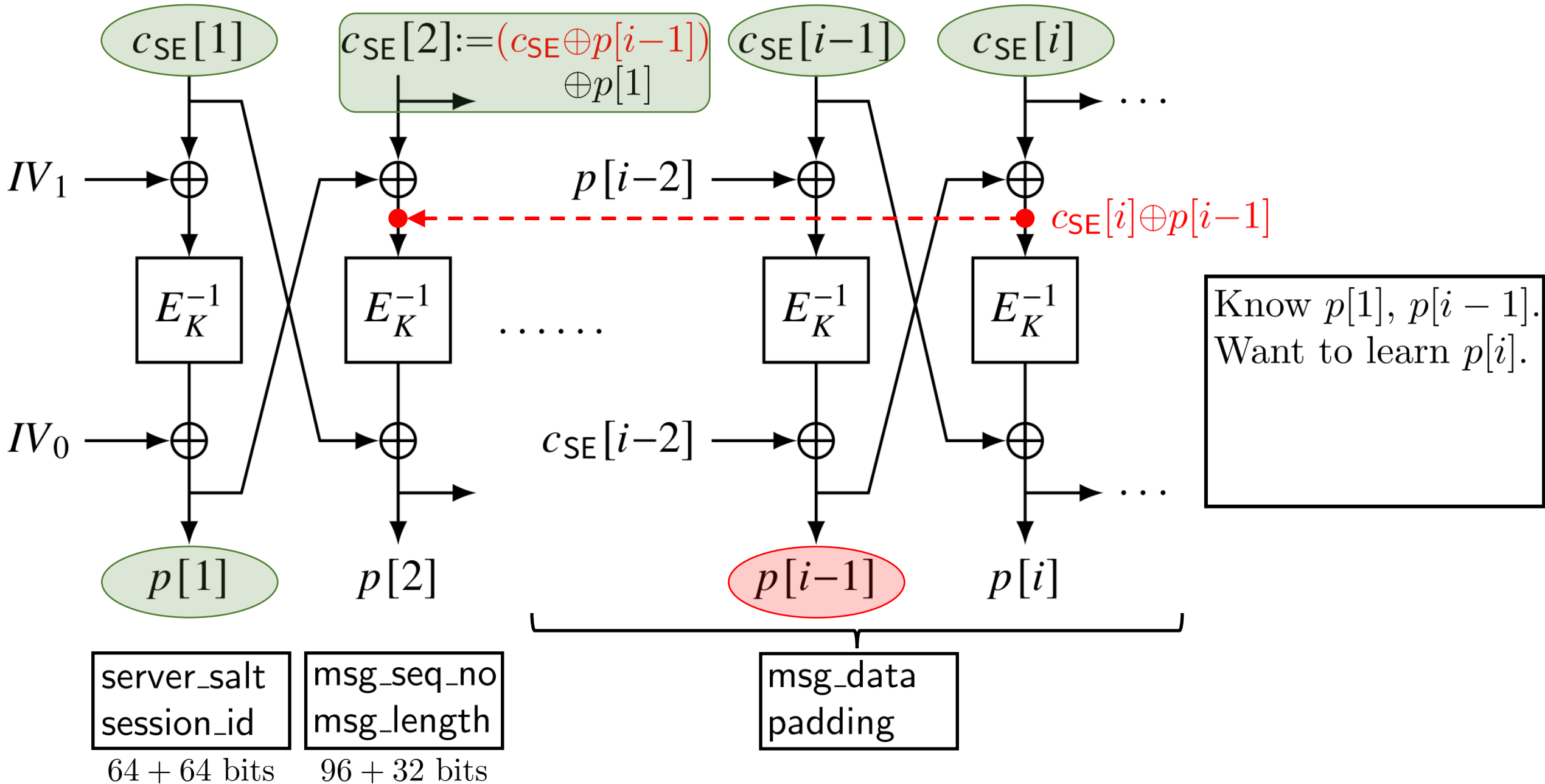
Timing Side-Channel Attacks



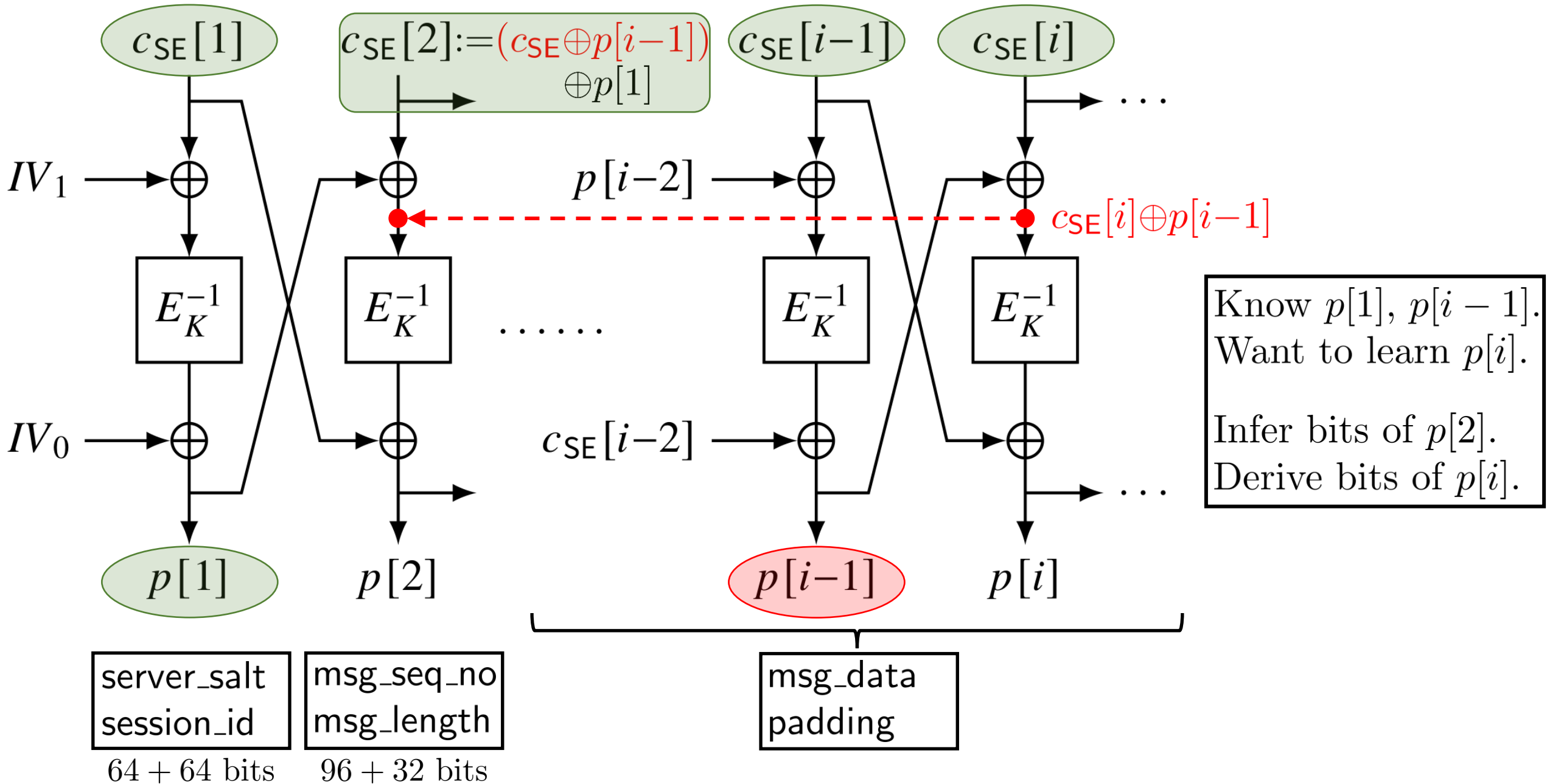
Timing Side-Channel Attacks



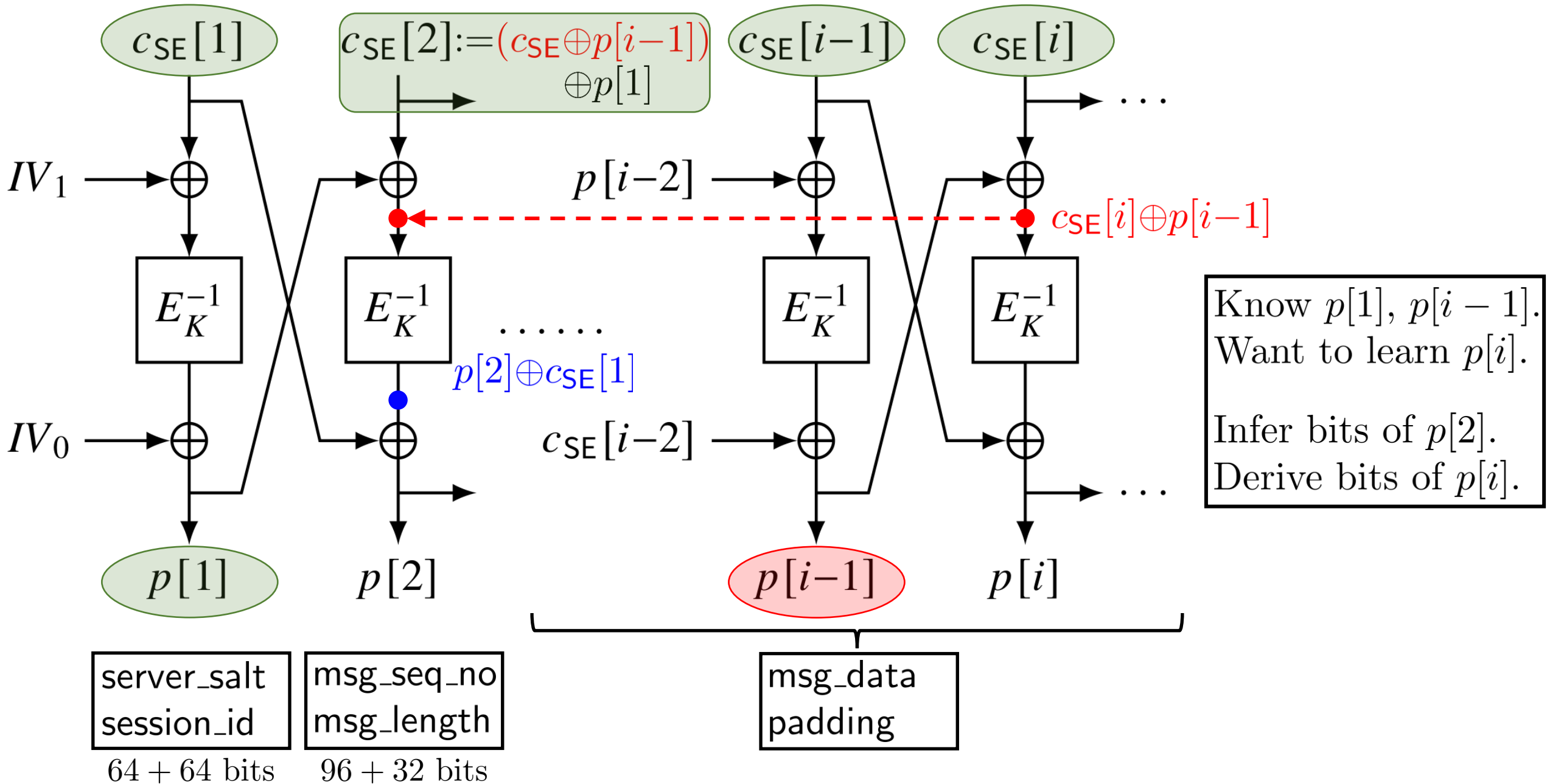
Timing Side-Channel Attacks



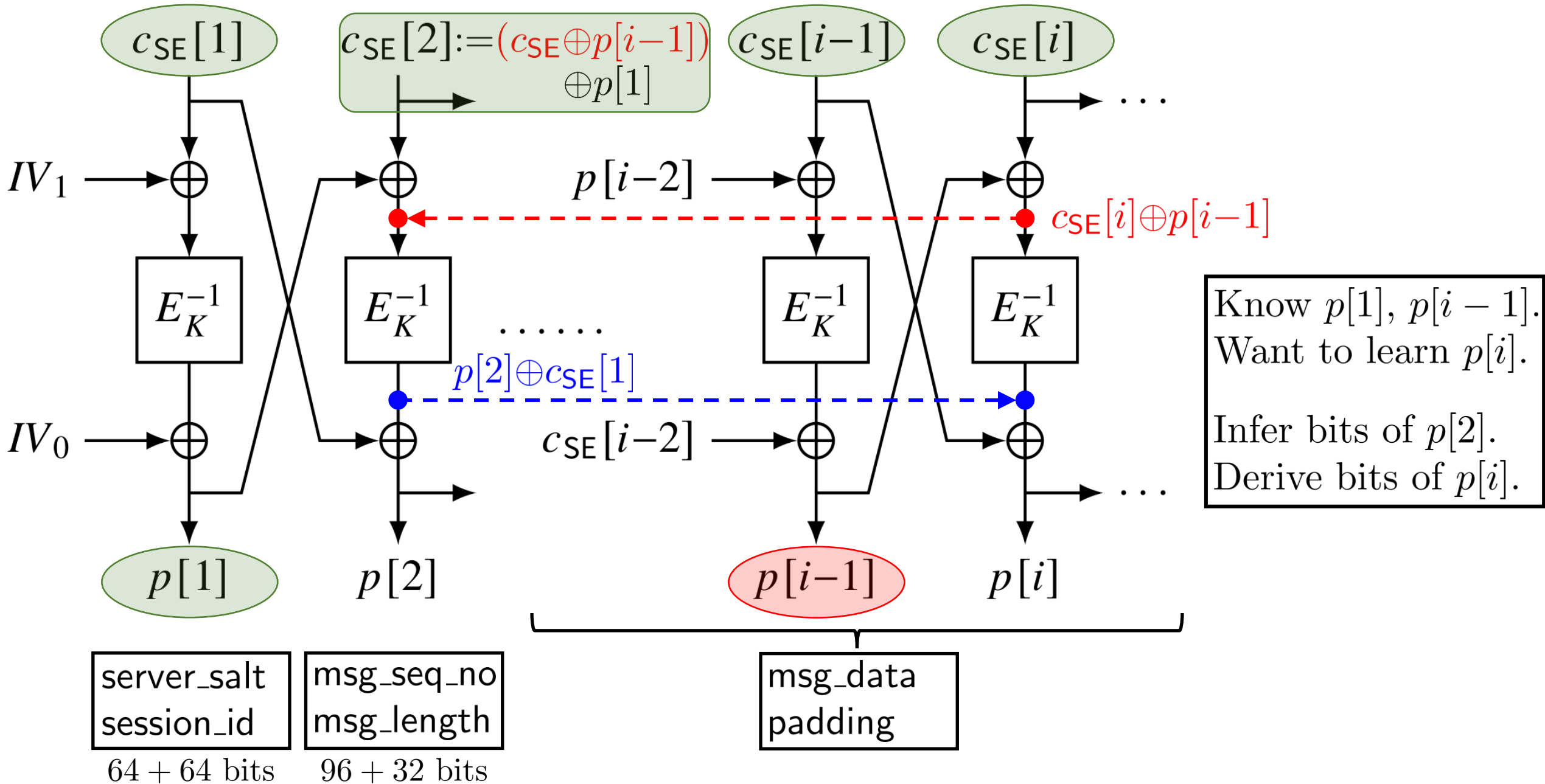
Timing Side-Channel Attacks



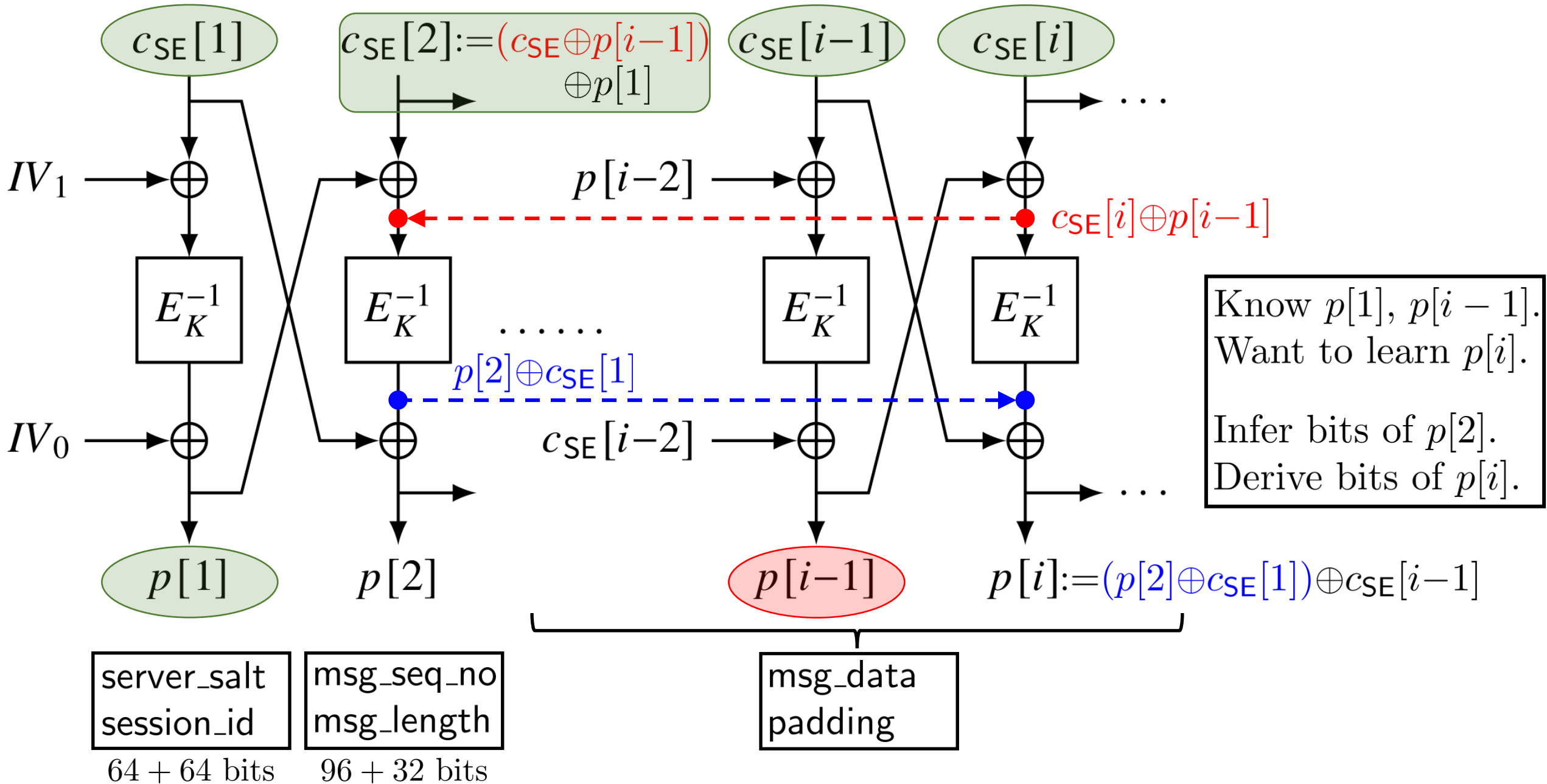
Timing Side-Channel Attacks



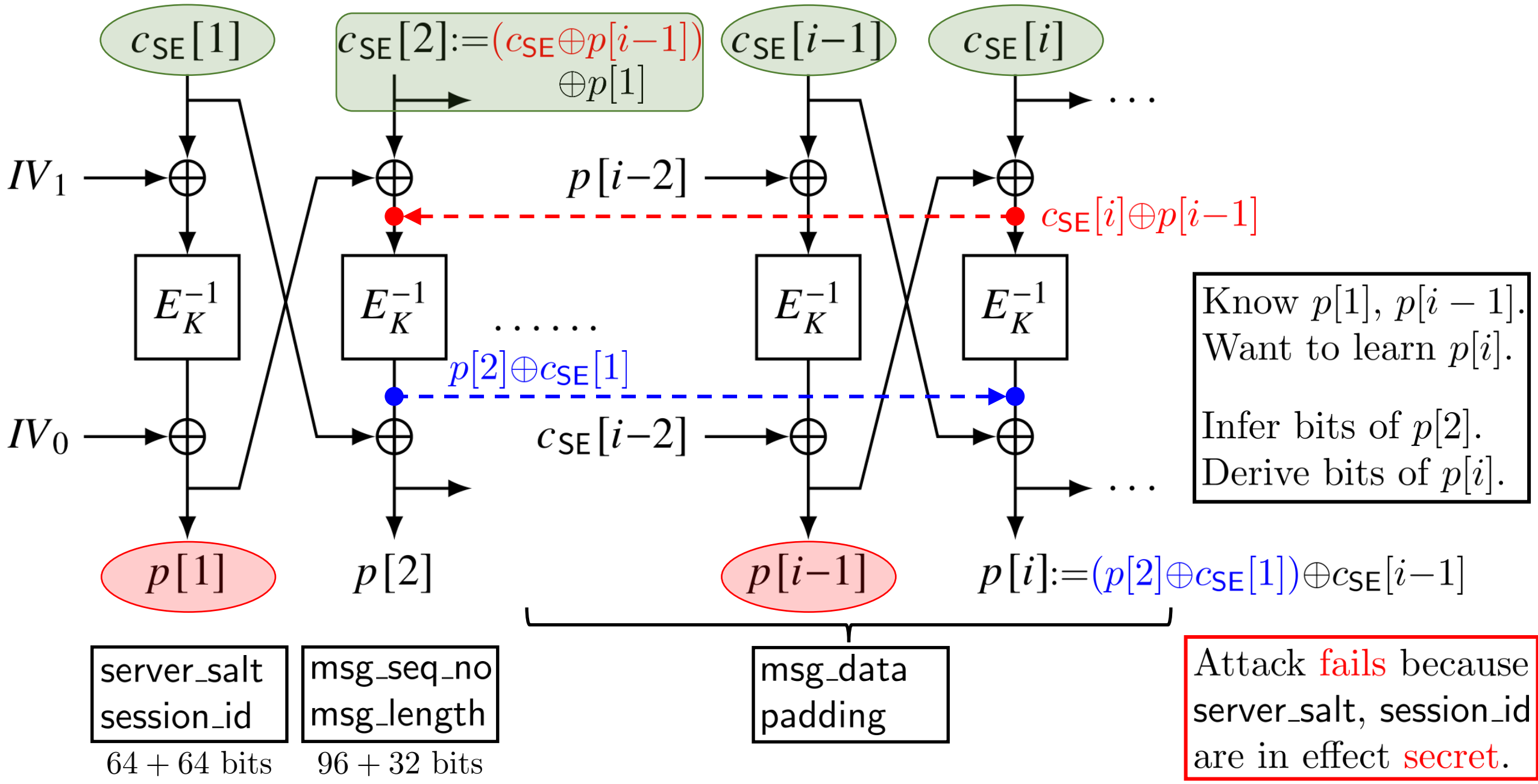
Timing Side-Channel Attacks



Timing Side-Channel Attacks

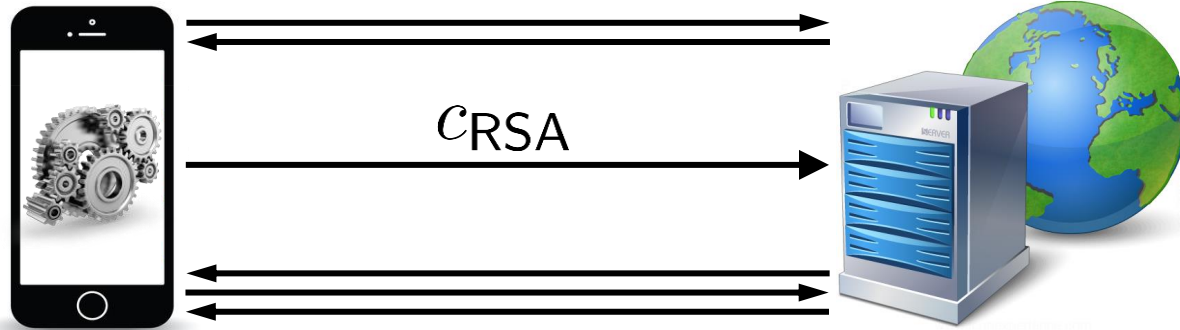


Timing Side-Channel Attacks



Timing Side-Channel Attack Against Servers

Telegram's key exchange

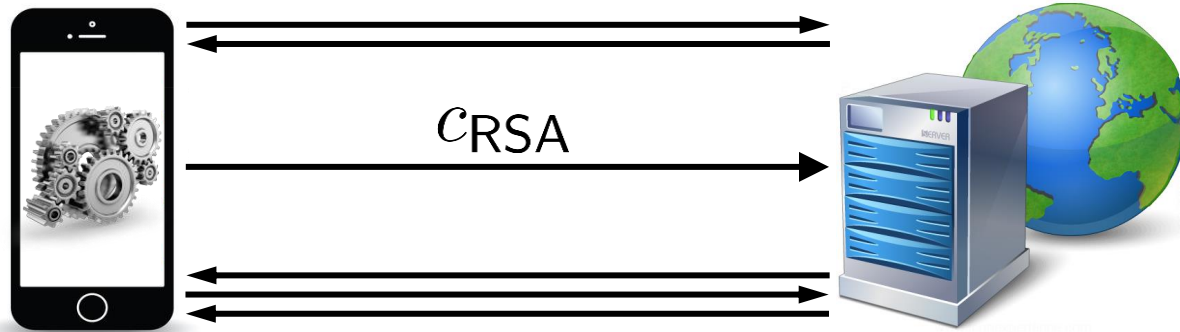


$$c_{RSA} \leftarrow \text{RSA.Enc}(pk, n)$$

$$n = \text{SHA-1}(\text{data}) \parallel \text{data} \parallel \text{padding}$$

Timing Side-Channel Attack Against Servers

Telegram's key exchange



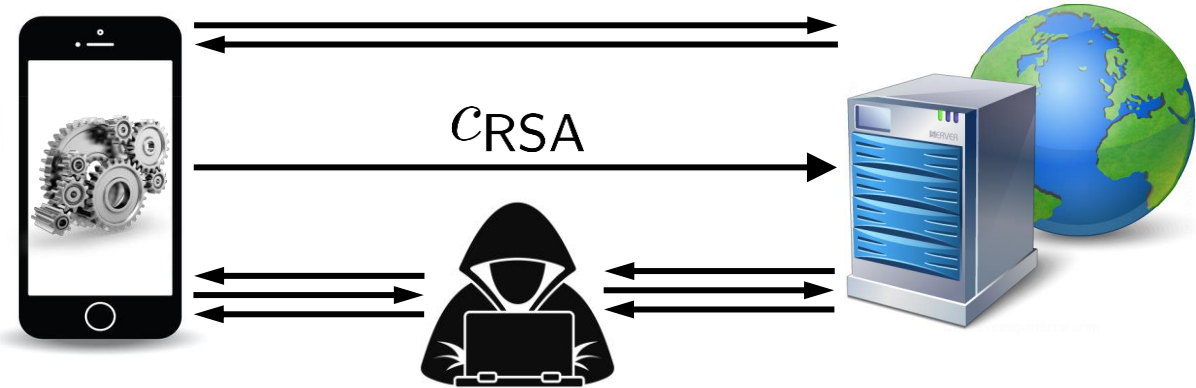
$$c_{RSA} \leftarrow \text{RSA.Enc}(pk, n)$$
$$n = \text{SHA-1}(\text{data}) \parallel \text{data} \parallel \text{padding}$$

If data can be recovered:

Attacker learns **server_salt** immediately.
Attacker learns **session_id** in $\approx 2^{64}$ queries.

The attack against IGE now works!

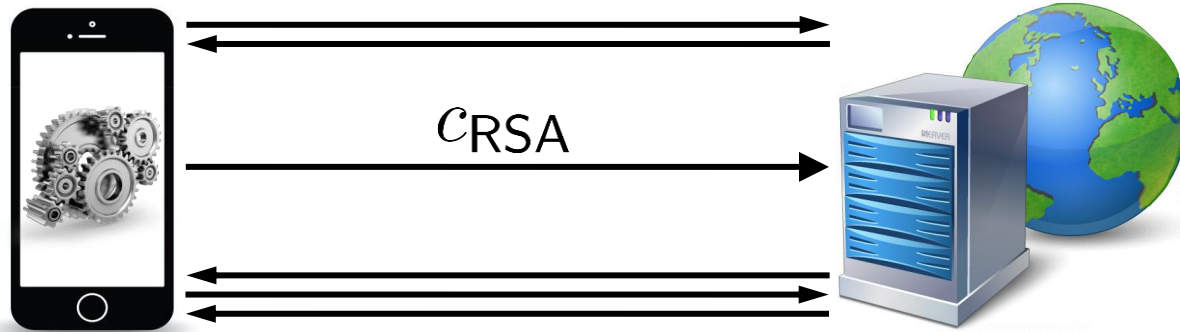
If data can be recovered **within 10 min**:



Man-in-the-middle attack.

Timing Side-Channel Attack Against Servers

Telegram's key exchange



$$c_{RSA} \leftarrow \text{RSA.Enc}(pk, n)$$

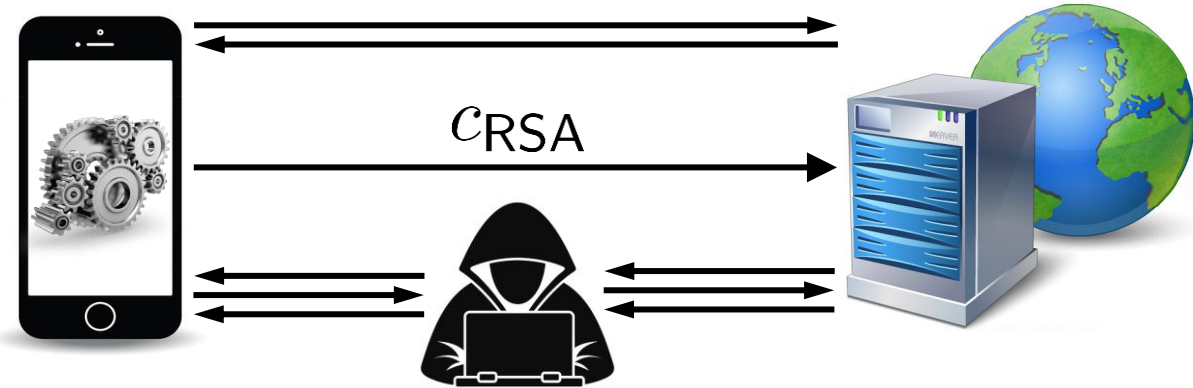
$$n = \text{SHA-1}(\text{data}) \parallel \text{data} \parallel \text{padding}$$

We **recover** data by solving noisy linear equations via **lattice reduction**.

If data can be recovered:

Attacker learns **server_salt** immediately.
Attacker learns **session_id** in $\approx 2^{64}$ queries.
The attack against IGE now works!

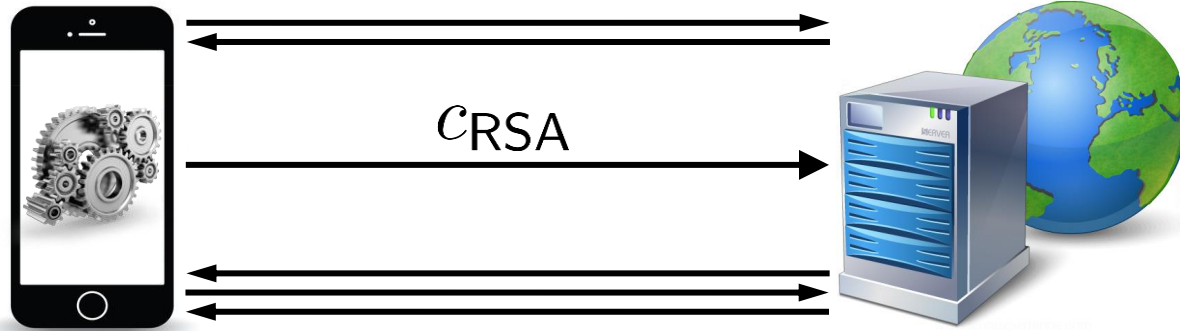
If data can be recovered **within 10 min**:



Man-in-the-middle attack.

Timing Side-Channel Attack Against Servers

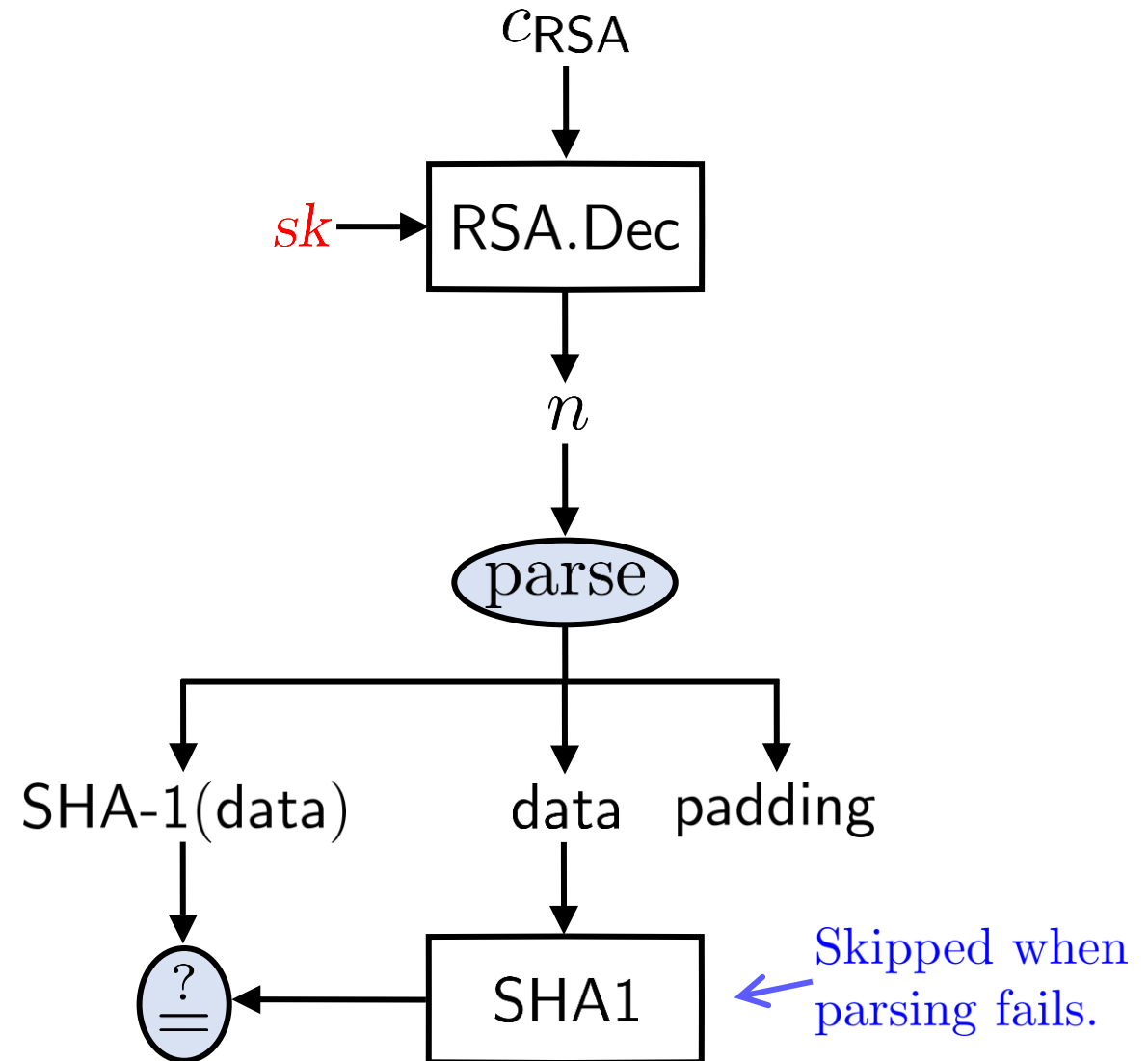
Telegram's key exchange



$$c_{RSA} \leftarrow \text{RSA.Enc}(pk, n)$$
$$n = \text{SHA-1}(\text{data}) \parallel \text{data} \parallel \text{padding}$$

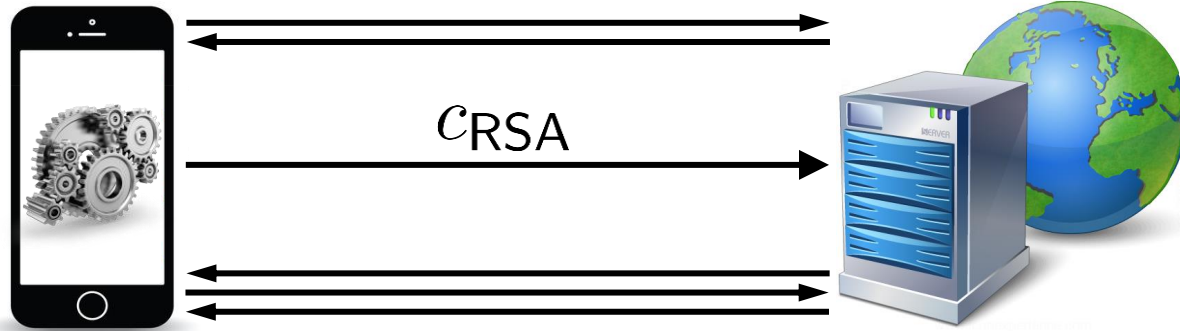
We **recover** data by solving noisy linear equations via **lattice reduction**.

Timing side-channel:



Timing Side-Channel Attack Against Servers

Telegram's key exchange



$$c_{RSA} \leftarrow \text{RSA.Enc}(pk, n)$$
$$n = \text{SHA-1}(\text{data}) \parallel \text{data} \parallel \text{padding}$$

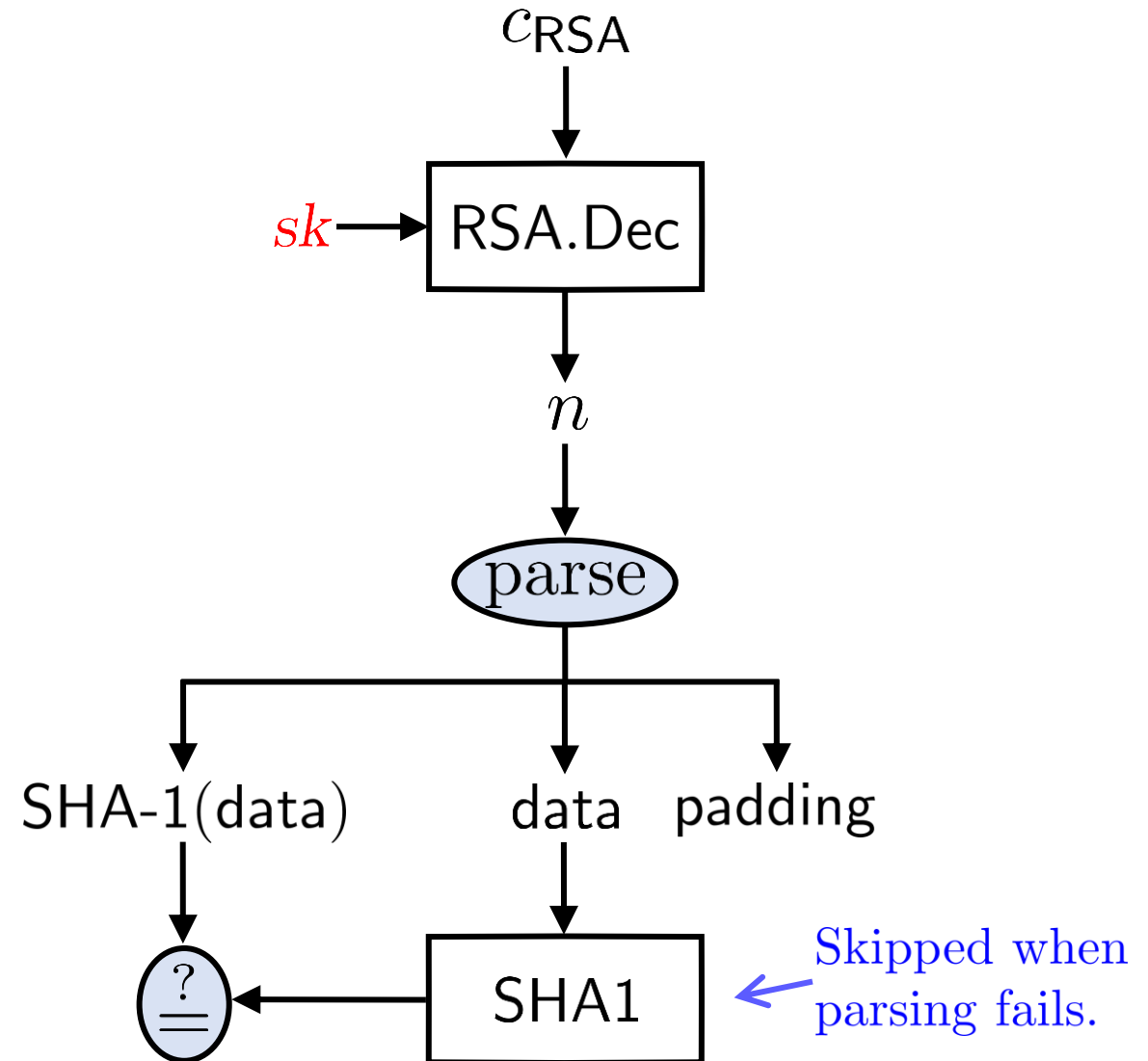
We **recover** data by solving noisy linear equations via **lattice reduction**.



Telegram uses **textbook** RSA scheme.

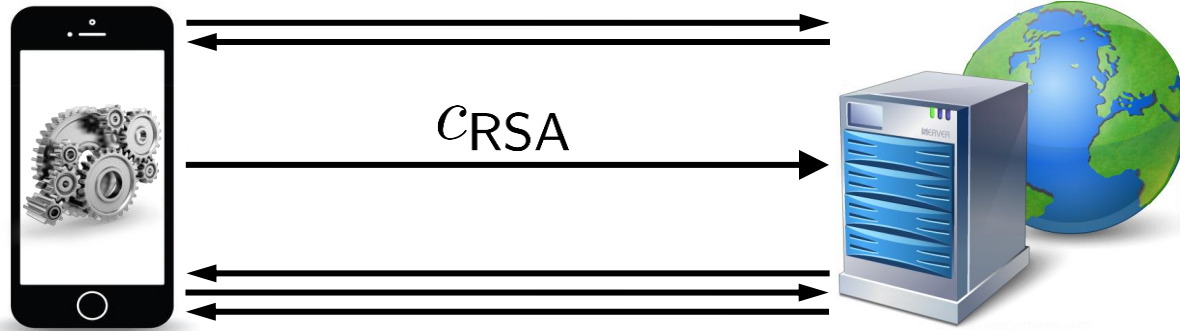
Why textbook RSA?

Timing side-channel:



Timing Side-Channel Attack Against Servers

Telegram's key exchange



$$c_{RSA} \leftarrow \text{RSA.Enc}(pk, n)$$
$$n = \text{SHA-1}(\text{data}) \parallel \text{data} \parallel \text{padding}$$

We **recover** data by solving noisy linear equations via **lattice reduction**.

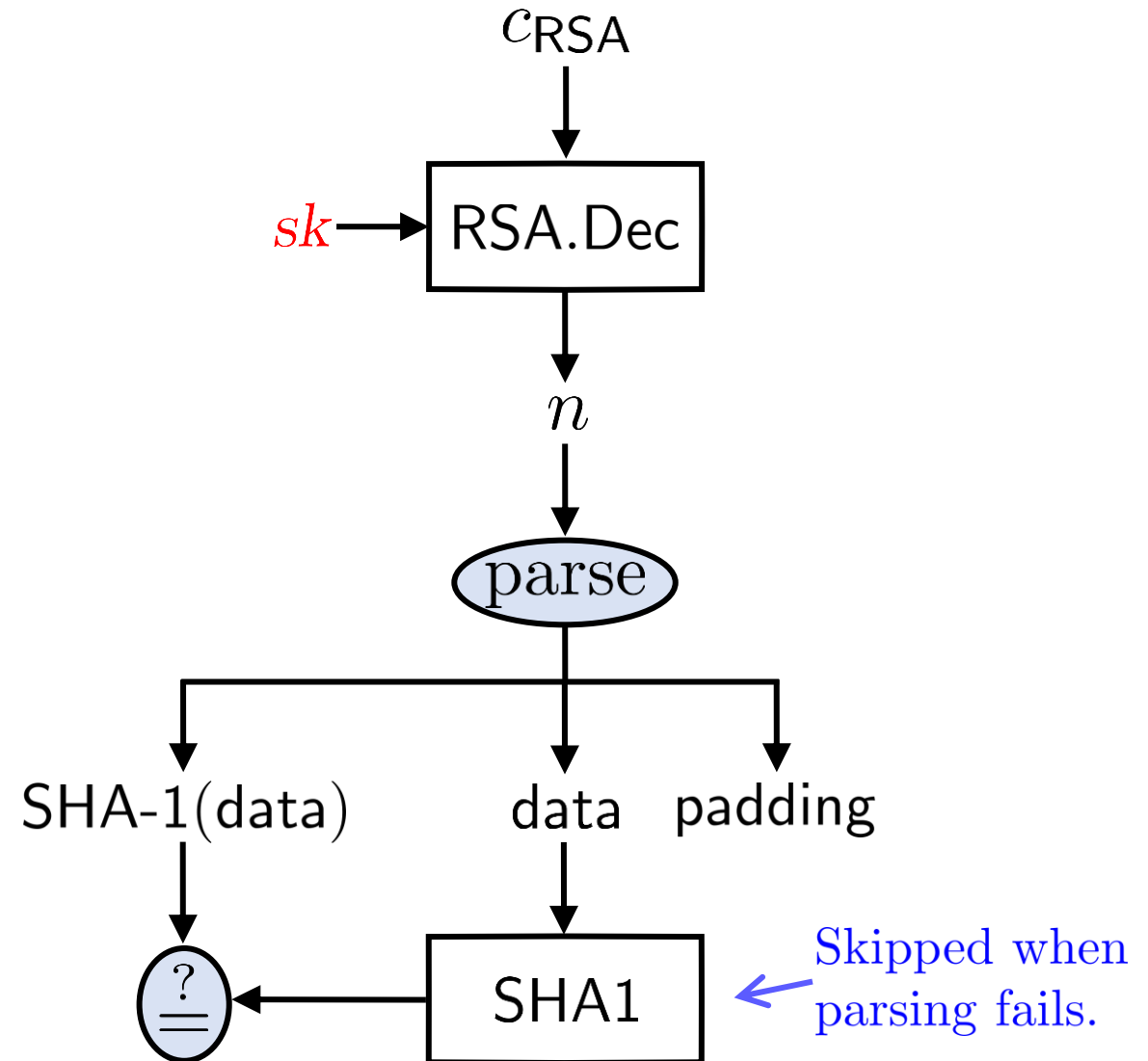


Telegram uses **textbook** RSA scheme.

Why textbook RSA?

Timings very small. **Infeasible** in practice.
Caveat: Telegram's server code is **secret**.

Timing side-channel:



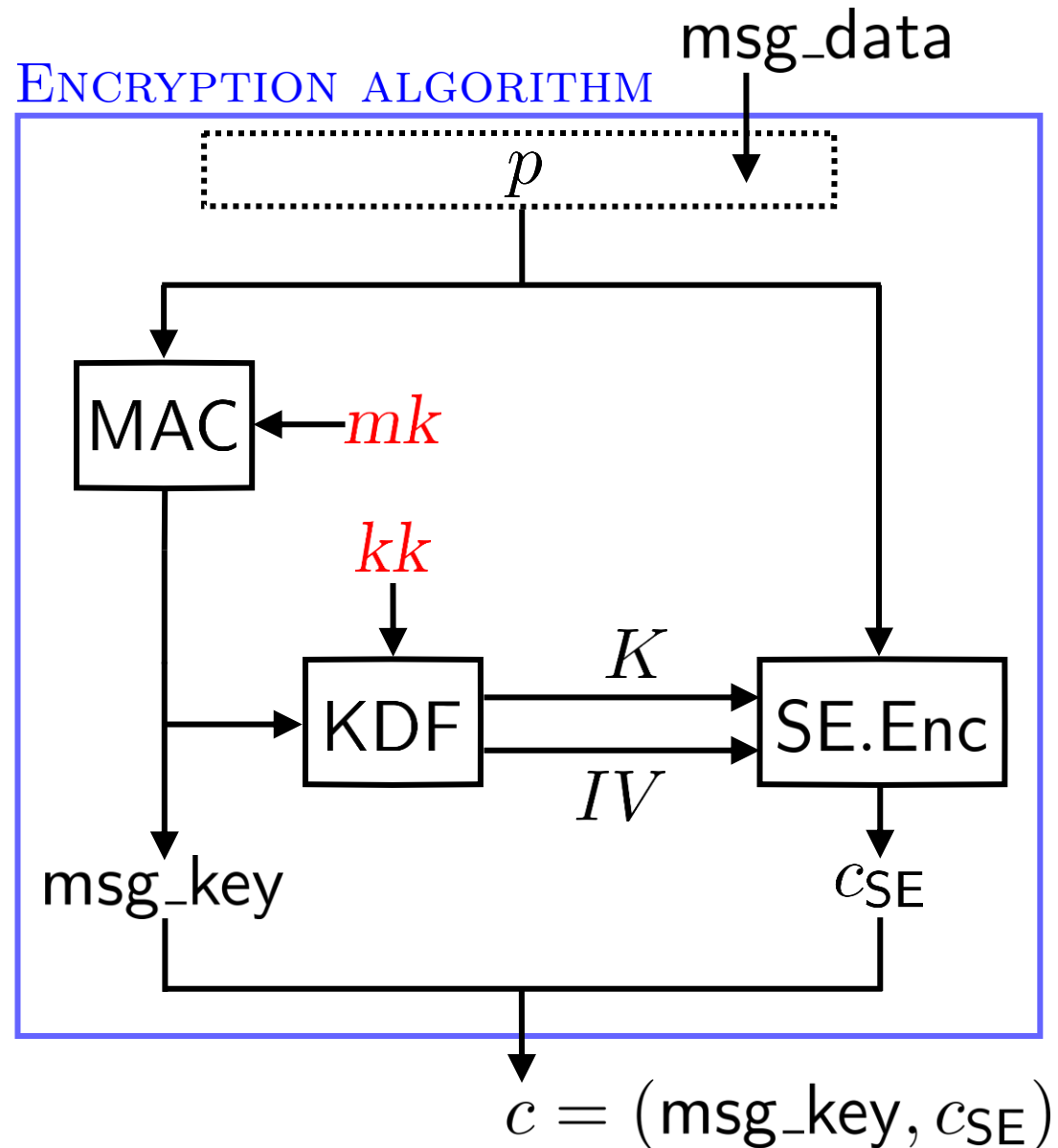
Attack Against IND-CPA Security

Theoretical attack.

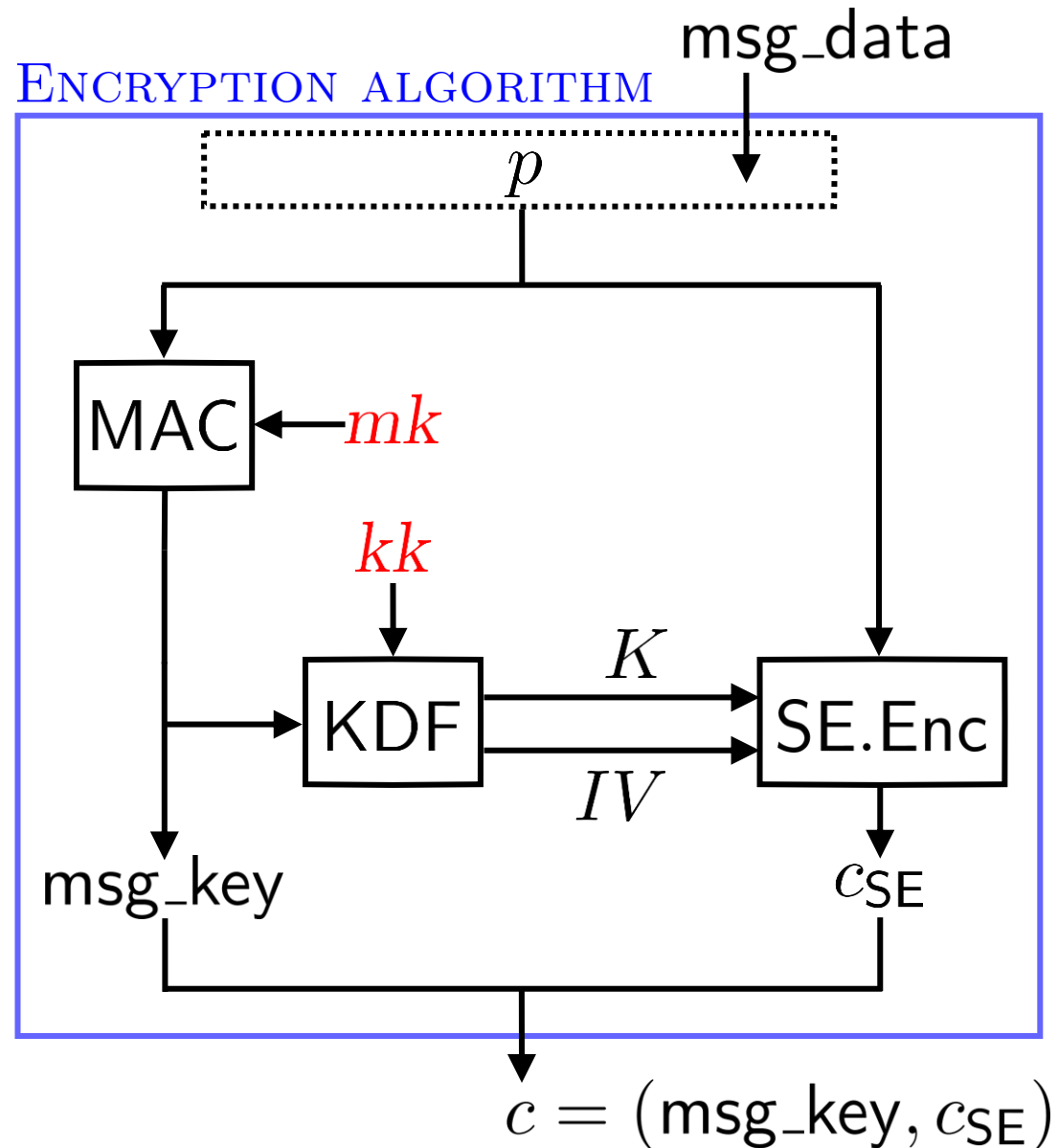
MTPProto requires message acknowledgements.
If no acknowledgement, then payload is resent.

Acknowledgements are **encrypted**.

Our attack subverts this.



Attack Against IND-CPA Security

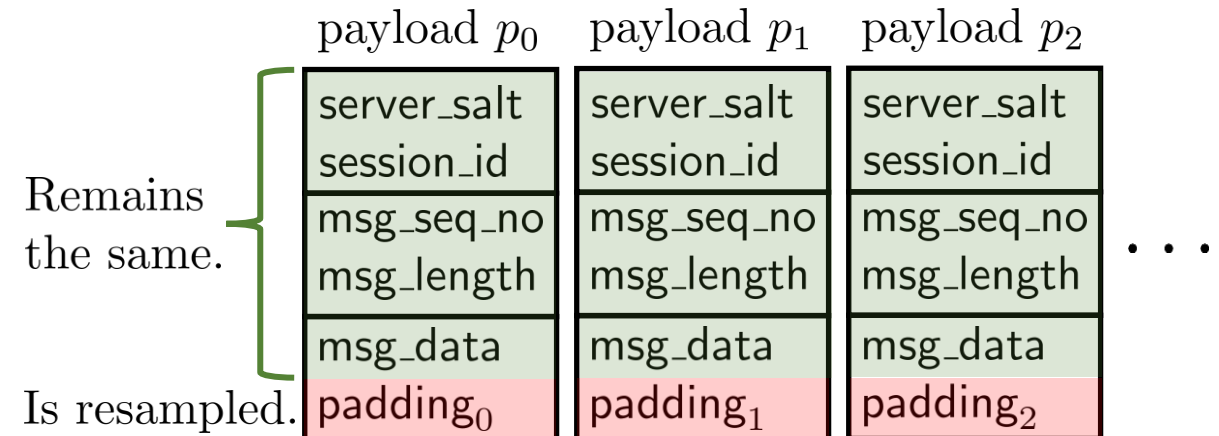


Theoretical attack.

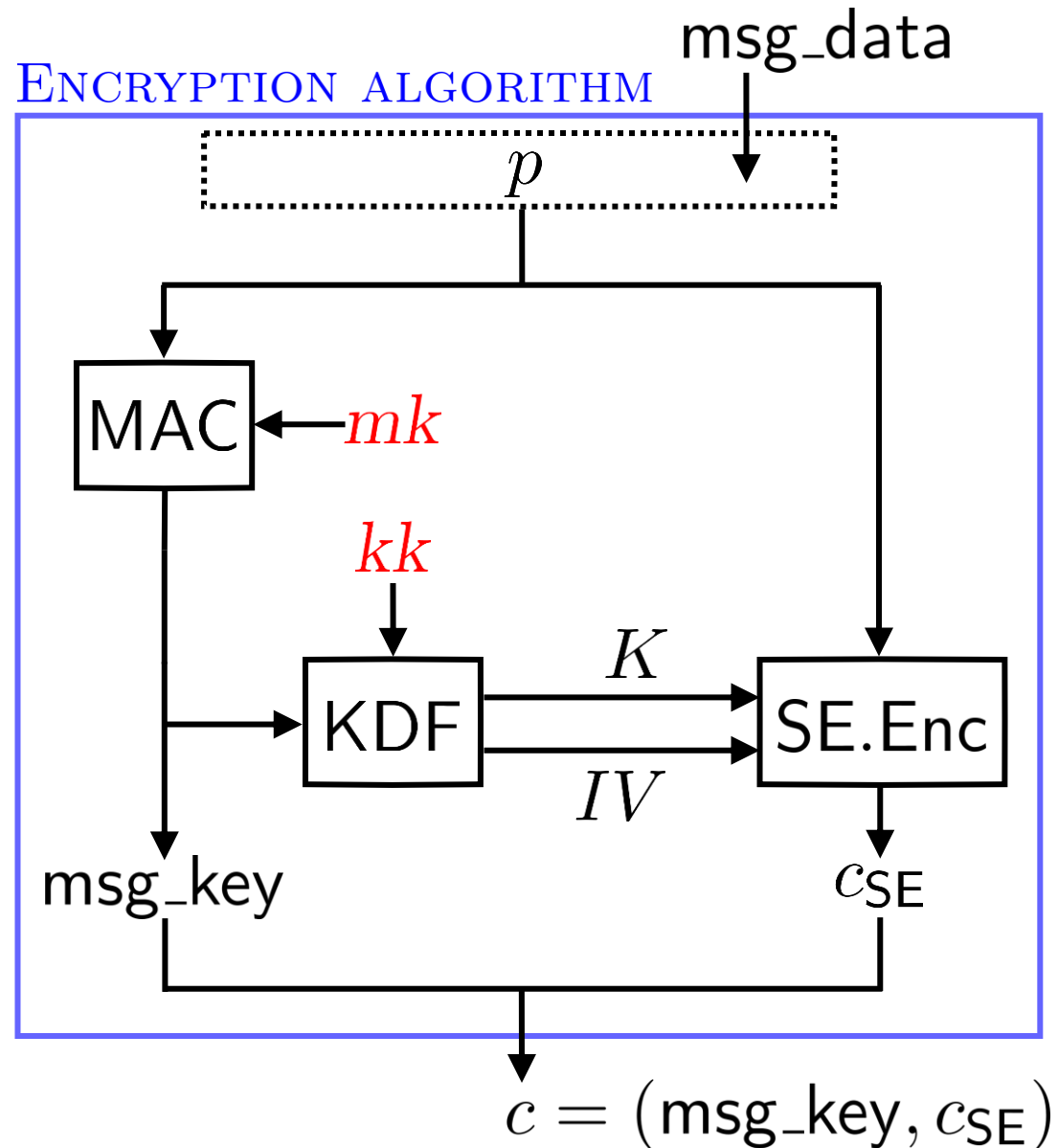
MTPProto requires message acknowledgements. If no acknowledgement, then payload is resent.

Acknowledgements are **encrypted**.

Our attack subverts this.



Attack Against IND-CPA Security

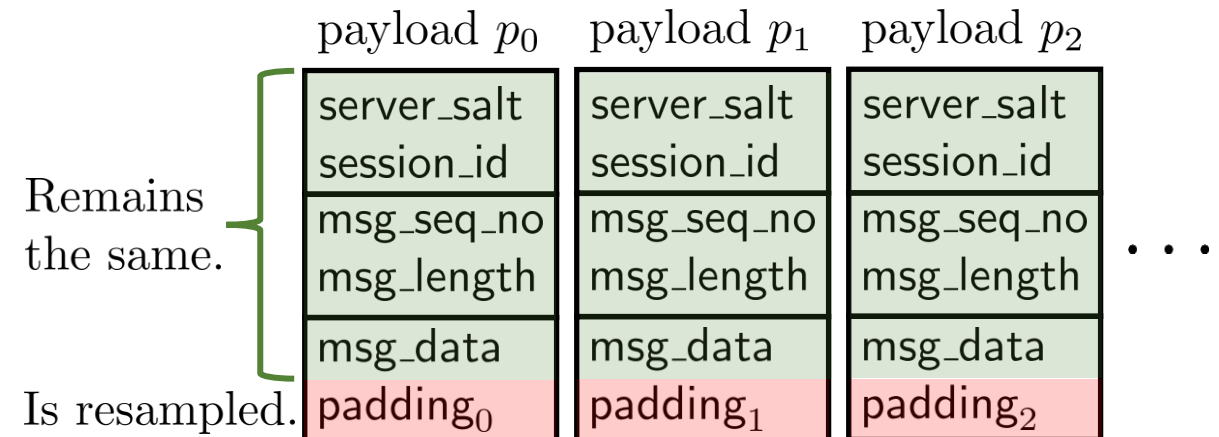


Theoretical attack.

MTPROTO requires message acknowledgements.
If no acknowledgement, then payload is resent.

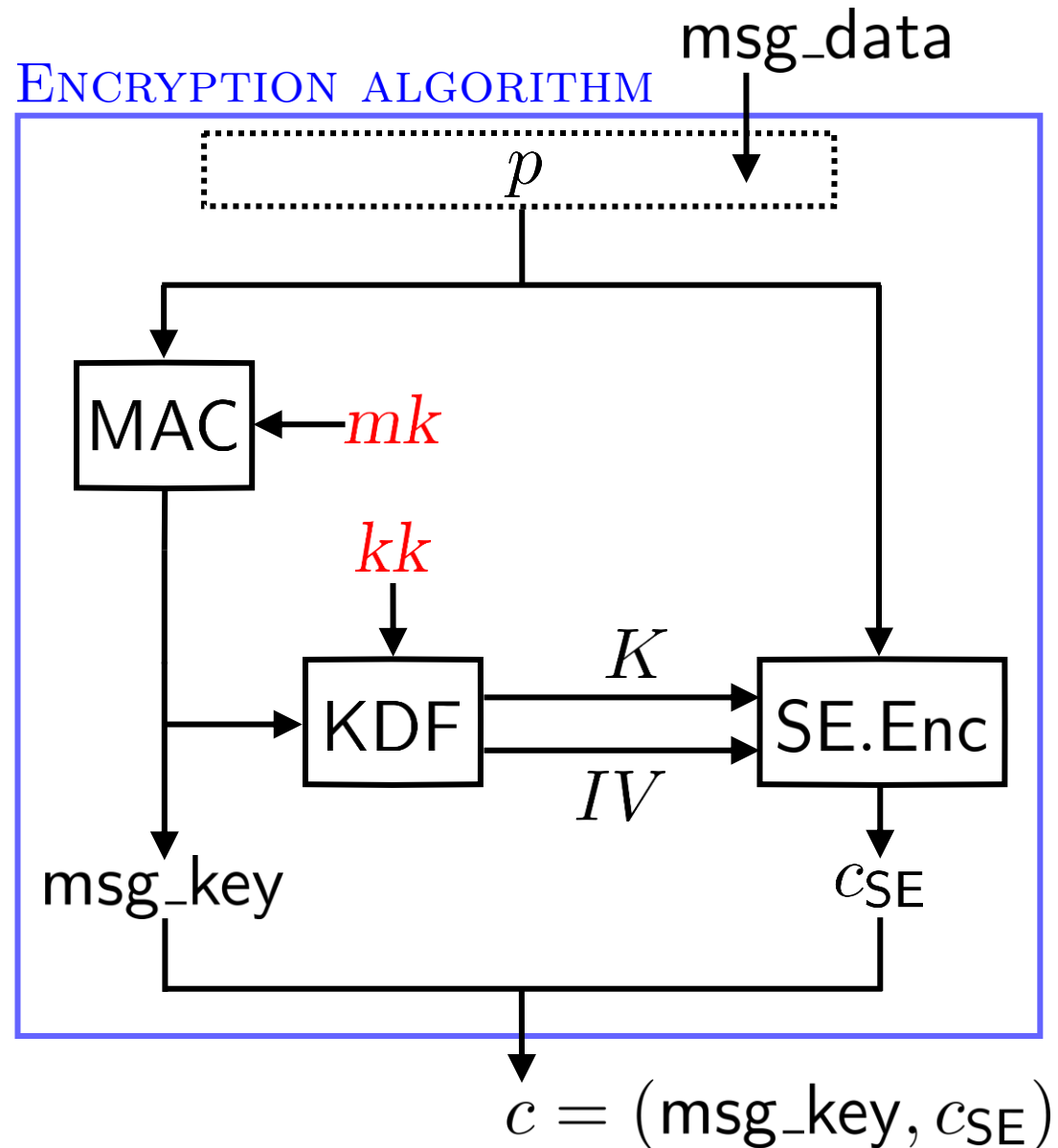
Acknowledgements are **encrypted**.

Our attack subverts this.



Observe $c = (\text{msg_key}, c_{SE})$
 $c^* = (\text{msg_key}^*, c_{SE}^*)$ with $\text{msg_key} = \text{msg_key}^*$
 $c_{SE}[2] = c_{SE}^*[2]$?

Attack Against IND-CPA Security

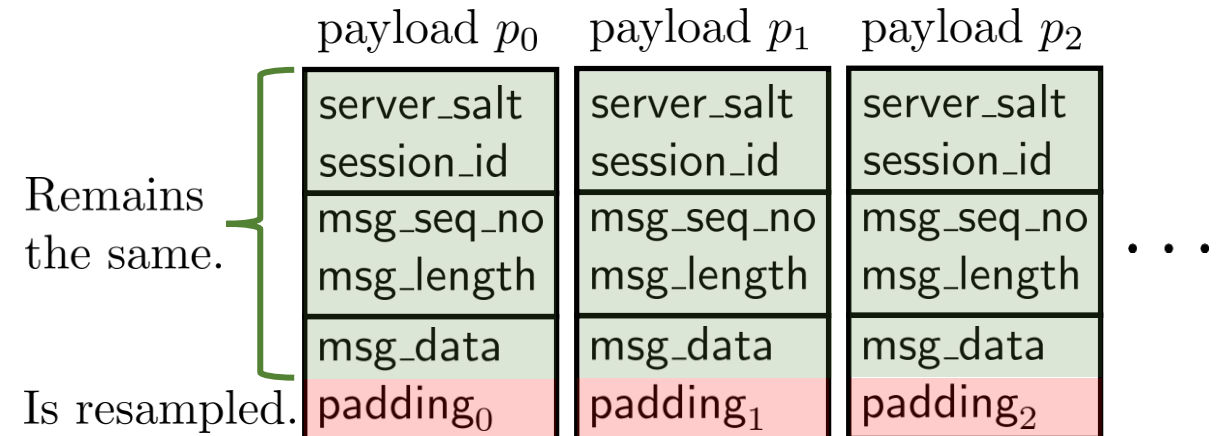


Theoretical attack.

MTPROTO requires message acknowledgements.
If no acknowledgement, then payload is resent.

Acknowledgements are **encrypted**.

Our attack subverts this.

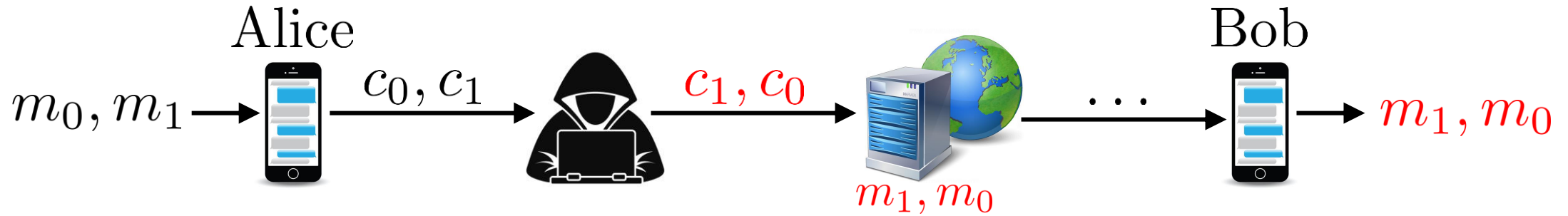
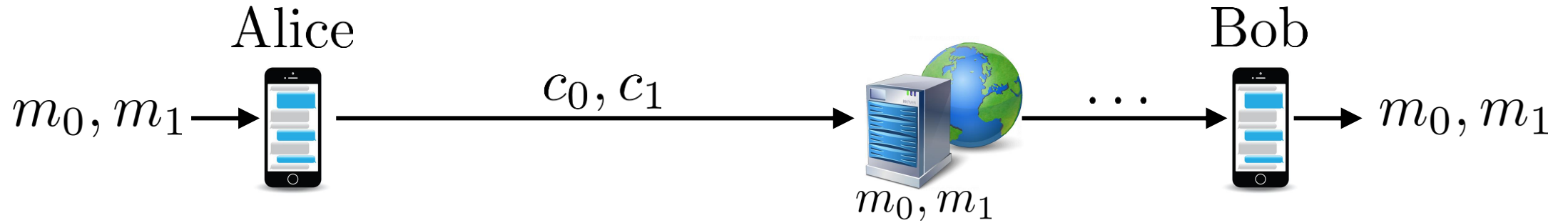


Observe $c = (\text{msg_key}, c_{SE})$ with $\text{msg_key} = \text{msg_key}^*$
 $c^* = (\text{msg_key}^*, c_{SE}^*)$ $c_{SE}[2] = c_{SE}^*[2]$?

Then c, c^* encrypt the same message!

No acknowledgement received between sending c, c^* .

Message Reordering Attack



Technically trivial. Easy to exploit.

Future Work

Large parts of **Telegram** **unstudied**:

SECRET CHATS
KEY EXCHANGE

..., multi-user security, forward secrecy,
Telegram Passport, bot APIs, higher-level
message processing, control messages, en-
crypted CDNs, cloud storage, ...



Thank you!

More information at
<https://mtpsym.github.io/>