A Concrete Treatment of Efficient Continuous Group Key Agreement via Multi-Recipient PKEs

Keitaro Hashimoto
Tokyo Tech/AIST, JP

Shuichi Katsumata
AIST, JP

Eamonn W. Postlethwaite
CWI, NL

Thomas Prest
PQShield SAS, FR

Bas Westerbaan
Cloudflare, NL

ACM CCS 2021

Outline

Efficient post-quantum CGKA protocol

1. Background
2. Our solution: Chained CmPKE
3. More efficient PQ multi-recipient PKEs
4. Comparison and implementation

Background: Secure (Group) Messaging

Secure (Group) Messaging

Recently, a lot of people use secure (group) messaging apps.

Applications	Num. of monthly active users
WhatsApp	2.0 billion
Facebook Messenger	1.3 billion
Telegram	550 million
Snapchat	514 million

Ref: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/

Secure (Group) Messaging

Recently, a lot of people use secure (group) messaging apps.

Applications	Num. of monthly active users
WhatsApp	2.0 billion
Facebook Messenger	1.3 billion
Telegram	550 million
Snapchat	514 million

Ref: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/

Because governments and hackers try to gather personal information.

- "NSA Prism program taps into user data of Apple, Google and others", The Guardian, 2013
  https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data
- "Al Jazeera journalists 'hacked via NSO Group spyware'", BBC, 2020
  https://www.bbc.com/news/technology-55396843
- "Grand jury subpoena for Signal user data, Central District of California", Signal , 2020
  https://signal.org/bigbrother/central-california-grand-jury/

Secure (Group) Messaging

Recently, a lot of people use secure (group) messaging apps.

Applications	Num. of monthly active users
WhatsApp	2.0 billion
Facebook Messenger	1.3 billion
Telegram	550 million
Snapchat	514 million

Ref: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/

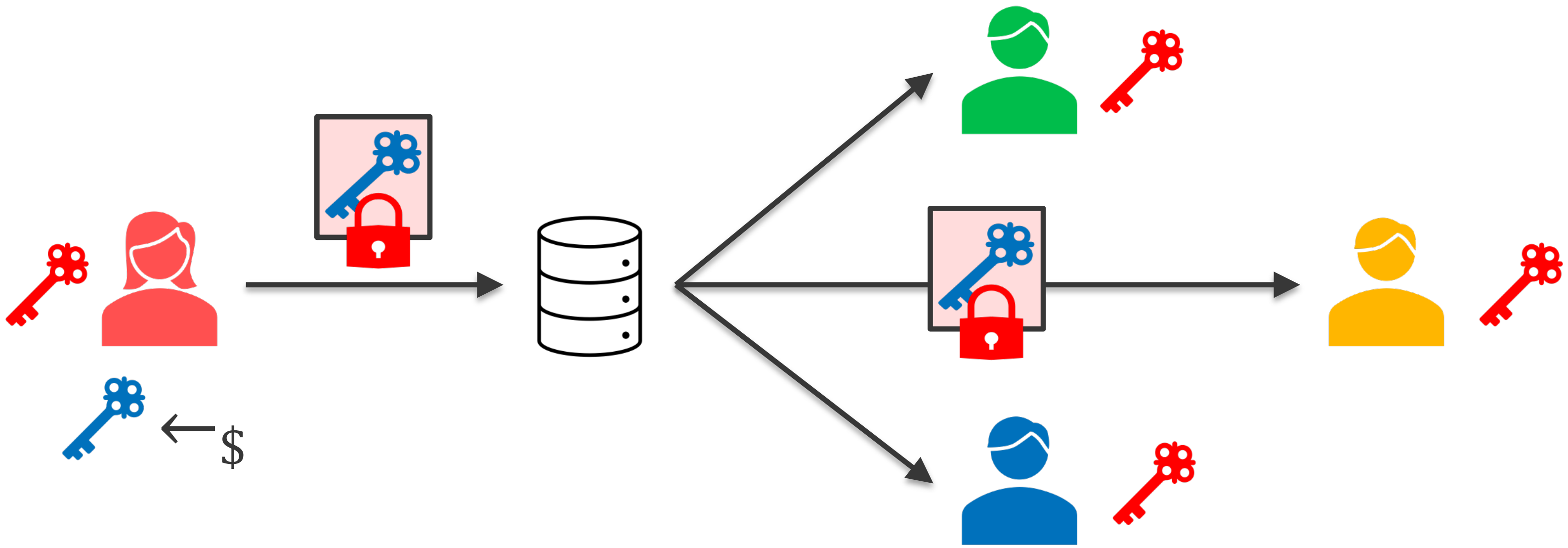
Existing secure (group) messaging:

- 2-party messaging: Signal protocol
  - Analyzed by a lot of works [CGC+17, ACD19, BFG+20, HKKP21]
- Group messaging : Continuous Group Key Agreement (this talk)

Continuous Group Key Agreement (CGKA) [ACDT20]

Group key agreement protocols that concentrate the cryptographic mechanisms of secure group messaging protocols:

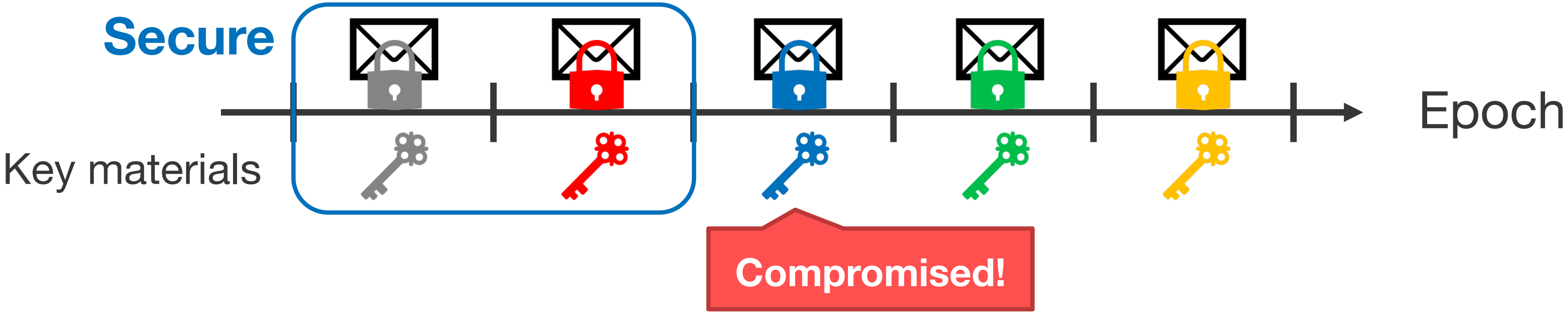
- Add a party to the group
- Remove a party from the group
- **Update key materials** (Ratcheting)



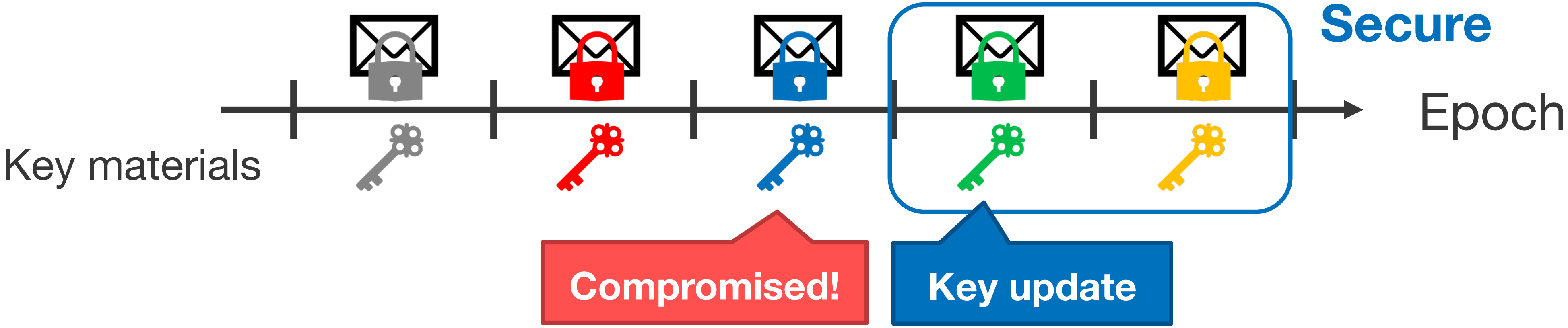
Continuous Group Key Agreement (CGKA) [ACDT20]

CGKA achieves strong security properties by updating key materials

- Forward secrecy (FS)



- Post-compromise security (PCS)



Existing CGKA protocols

- TreeKEM [BBR18, BBN19, ACDT20, ACJM20, AJM20…]
  - Used in IETF Messaging Layer Security (MLS) [OBR+21, BBM+20]
- Chained mKEM [BBN19]
  - Based on multi-recipient PKE (mPKE)
  - Starting point of our study

Bandwidth cost for key update (N: group size)

Scheme	Upload cost	Download cost	Total cost (upload + N-1 download)
TreeKEM	$\Omega(\log N)$	$\Omega(\log N)$	$\Omega(N \log N)$
Chained mKEM	$O(N)$	$O(N)$	$O(N^2)$

Efficient key update is important

As the group size N increases,

- the size of key update messages also increases
- the frequency of key update also increases
  - Likelihood of key compromise is higher for large group

Efficient key update is important

As the group size N increases,

- the size of key update messages also increases
- the frequency of key update also increases
  - Likelihood of key compromise is higher for large group

This tension is amplified by two factors:

- Messaging apps target mobile devices
  - Data cap per month is limited (e.g., 1GB)

- Post-quantum cryptography
  - Consume x10 or more bandwidth than classical counterpart
  - Example: TreeKEM with Classic McEliece [ABC+20] used in 256 users' group.
    If each user updates its key material twice, it costs 1 GB for each user.

Efficient key update is important

As the group size N increases,

- the <u>size</u> of key update messages also increases
- the <u>frequency</u> of key update also increases
  - Likelihood of key compromise is higher for large group

This tension is amplified by two factors:

- Messaging apps target <u>mobile devices</u>
  - Data cap per month is limited (e.g., 1GB)

- Post-quantum cryptography
  - Consume x10 or more bandwidth than classical counterpart
  - Example: TreeKEM with Classic McEliece [ABC+20] used in 256 users' group. If each user updates its key material twice, it costs 1 GB for each user.

Smaller key update costs are desirable in the real-world!

Efficient key update is important

As the group size N increases,

- the size of key update messages also increases
- the frequency of key update also increases
  - Likelihood of key compromise is higher for large group

This tension is amplified by two factors:

- Messaging apps target mobile devices
  - Data cap per month is limited (e.g., 1GB)
- Post-quantum cryptography
  - Consume x10 or more bandwidth than classical counterpart

Purpose

Design PQ CGKA protocol with small key update costs

Our contribution

Chained CmPKE: CGKA with asymmetric bandwidth cost

Scheme	Upload cost	Download cost	Total cost (upload + N-1 download)
TreeKEM	$\Omega(\log N)$	$\Omega(\log N)$	$\Omega(N \log N)$
Chained mKEM	$O(N)$	$O(N)$	$O(N^2)$
Chained CmPKE	$O(N)^\star$	$\boldsymbol{O(1)}$	$\boldsymbol{O(N)}$

$\star$: When N is about hundreds, the concrete upload cost is smaller than TreeKEM.

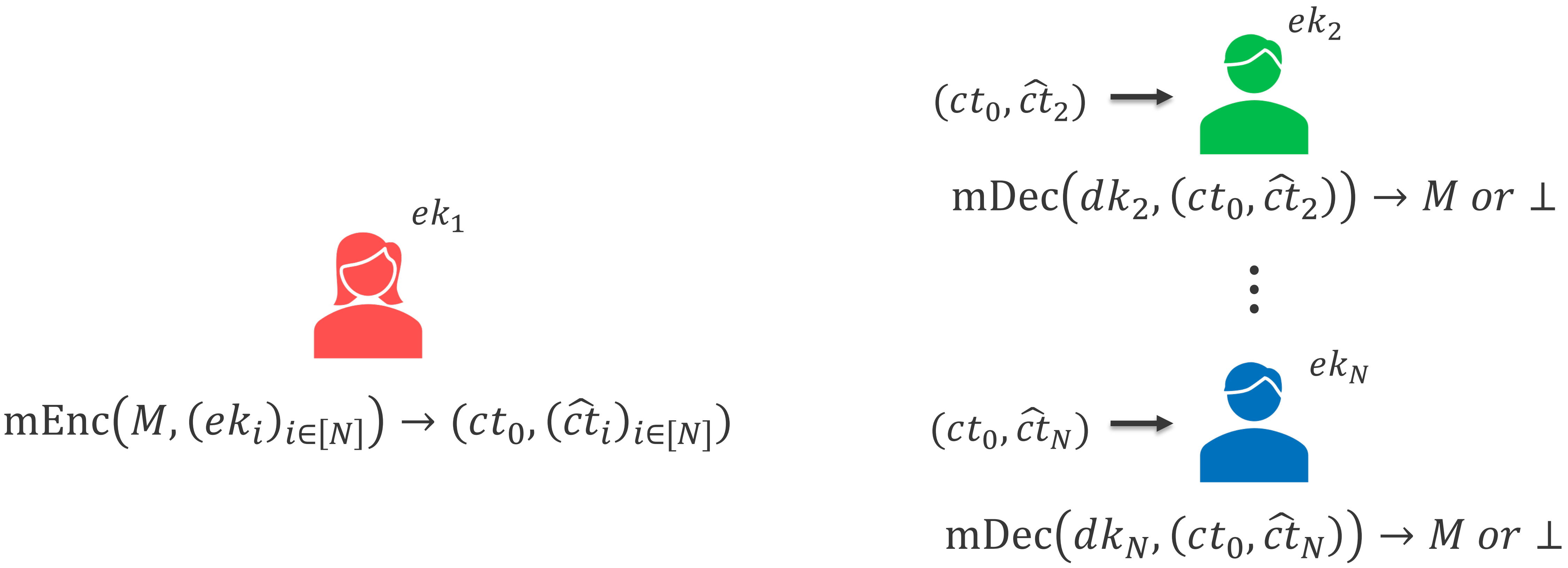
Chained CmPKE is based on Chained mKEM with two new ideas:

1. **Committing mPKE** ⇒ achieve $O(1)$ download cost
2. **More efficient PQ mPKE** ⇒ reduce the concrete size of key update messages

Contribution 1

New CGKA: Chained CmPKE

Racap: Multi-recipient PKE (mPKE)

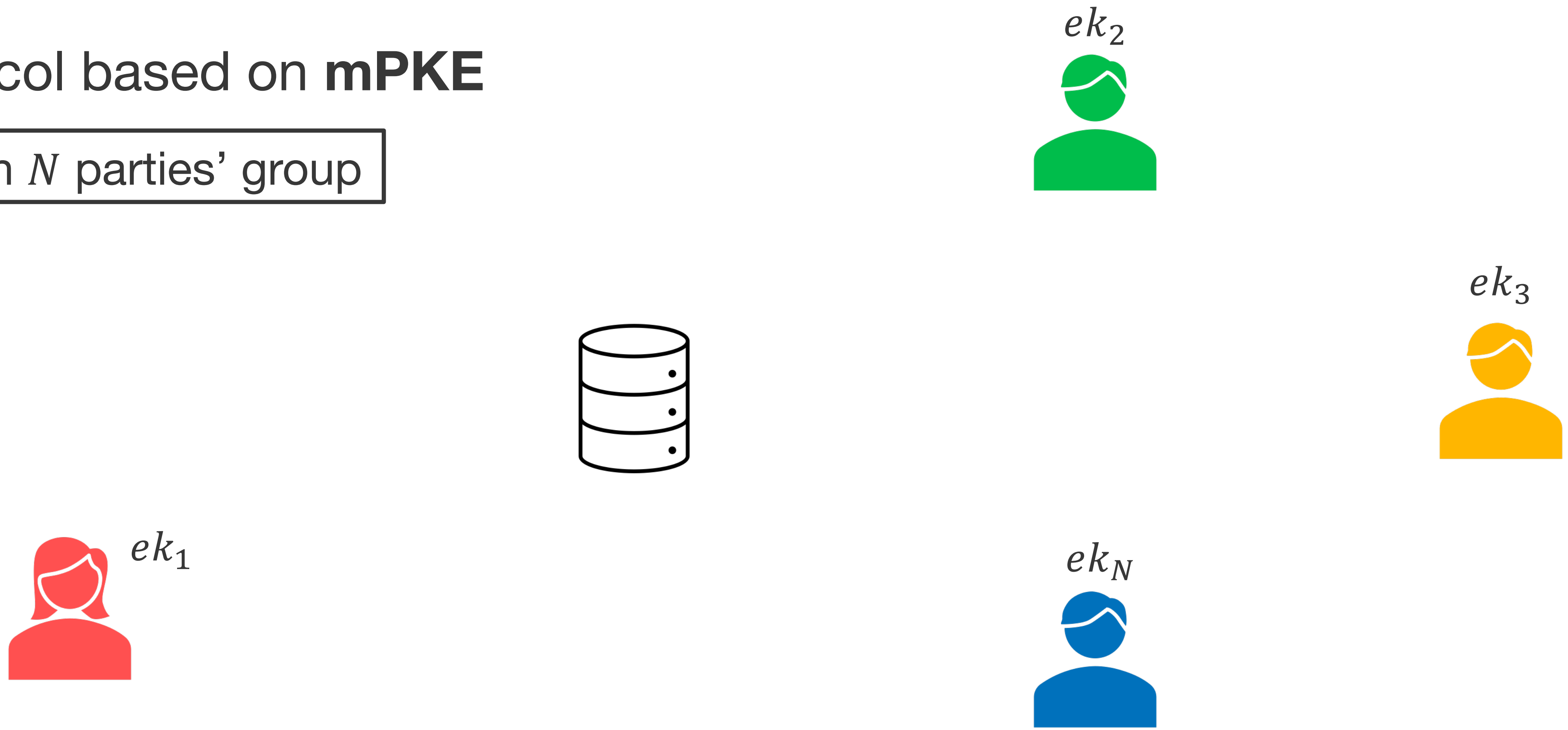


- The same message M can be efficiently encrypted to N parties
- Recently, [KKPP20] has revisited mPKE in the post-quantum setting
  - $|\widehat{ct}_i| \ll |ct_0|$ in this setting

Starting point: Chained mKEM [BBN19]

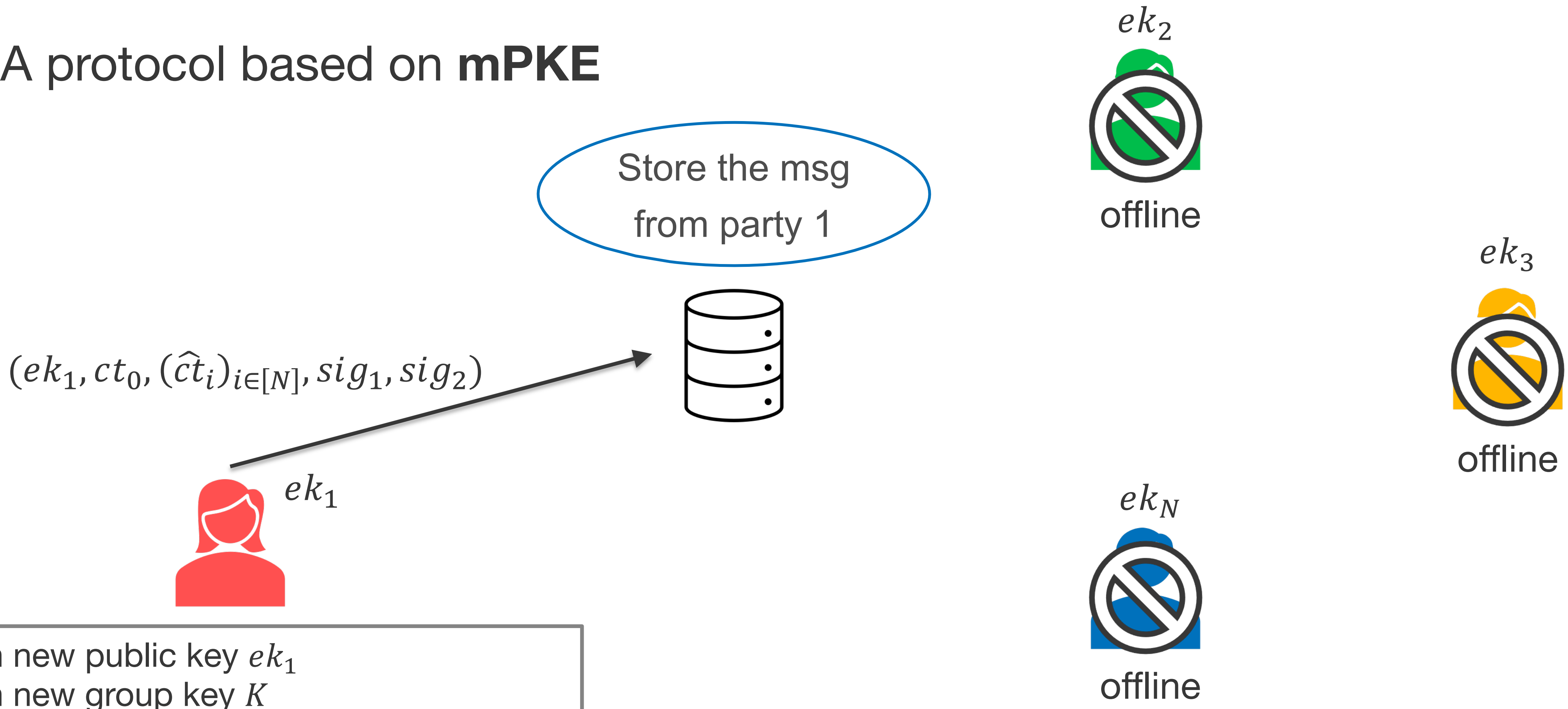
CGKA protocol based on **mPKE**

Key update on N parties' group



Chained mKEM [BBN19]

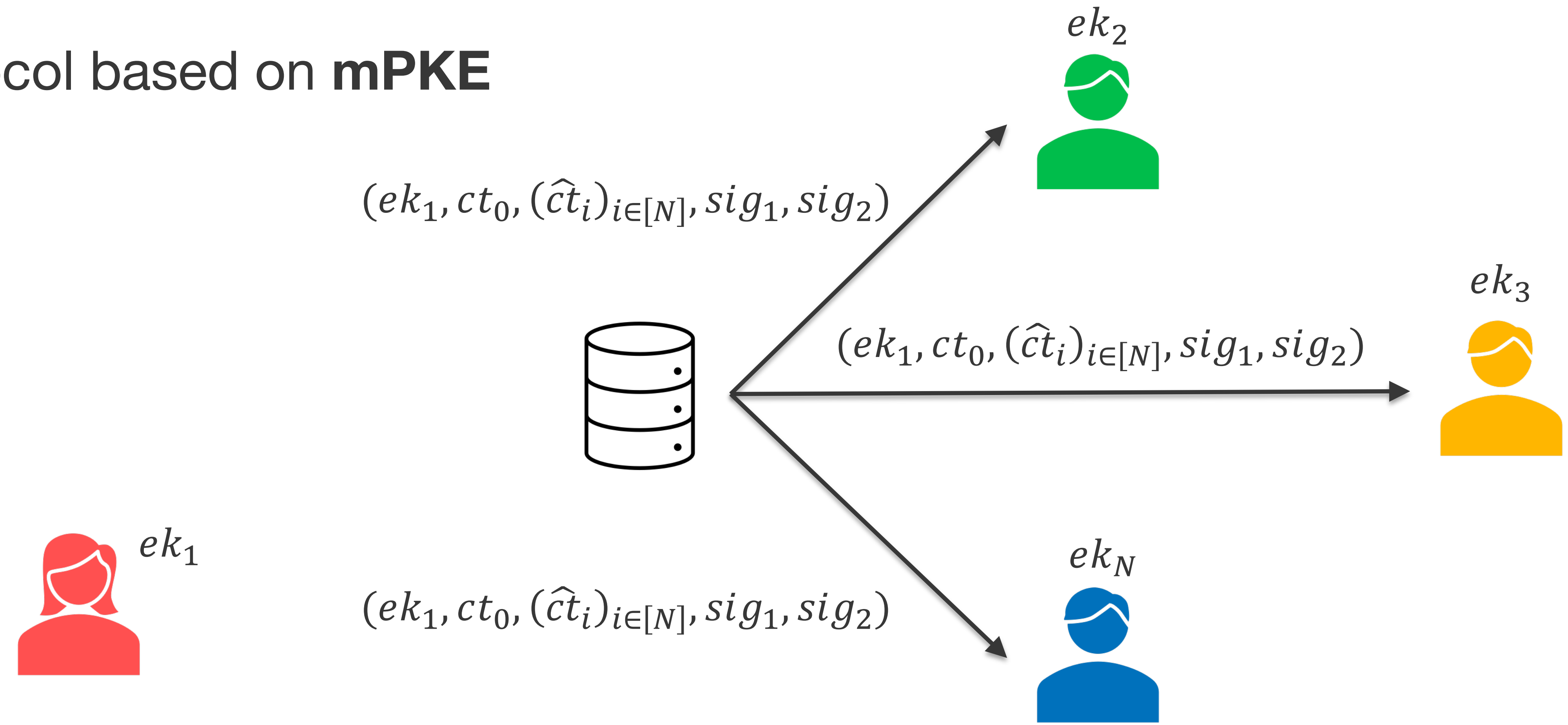
CGKA protocol based on **mPKE**



1. Gen new public key ek_1
2. Gen new group key K
3. Gen $\left(ct_0, (\hat{ct}_i)_{i\in[N]}\right) \leftarrow \text{mEnc}(K, (ek_i)_{i\in[N]})$
4. Gen $sig_1 \leftarrow \text{Sign}(sk_1, ek_1)$
5. Gen $sig_2 \leftarrow \text{Sign}\left(sk_1, (ct_0, (\hat{ct}_i)_{i\in[N]})\right)$

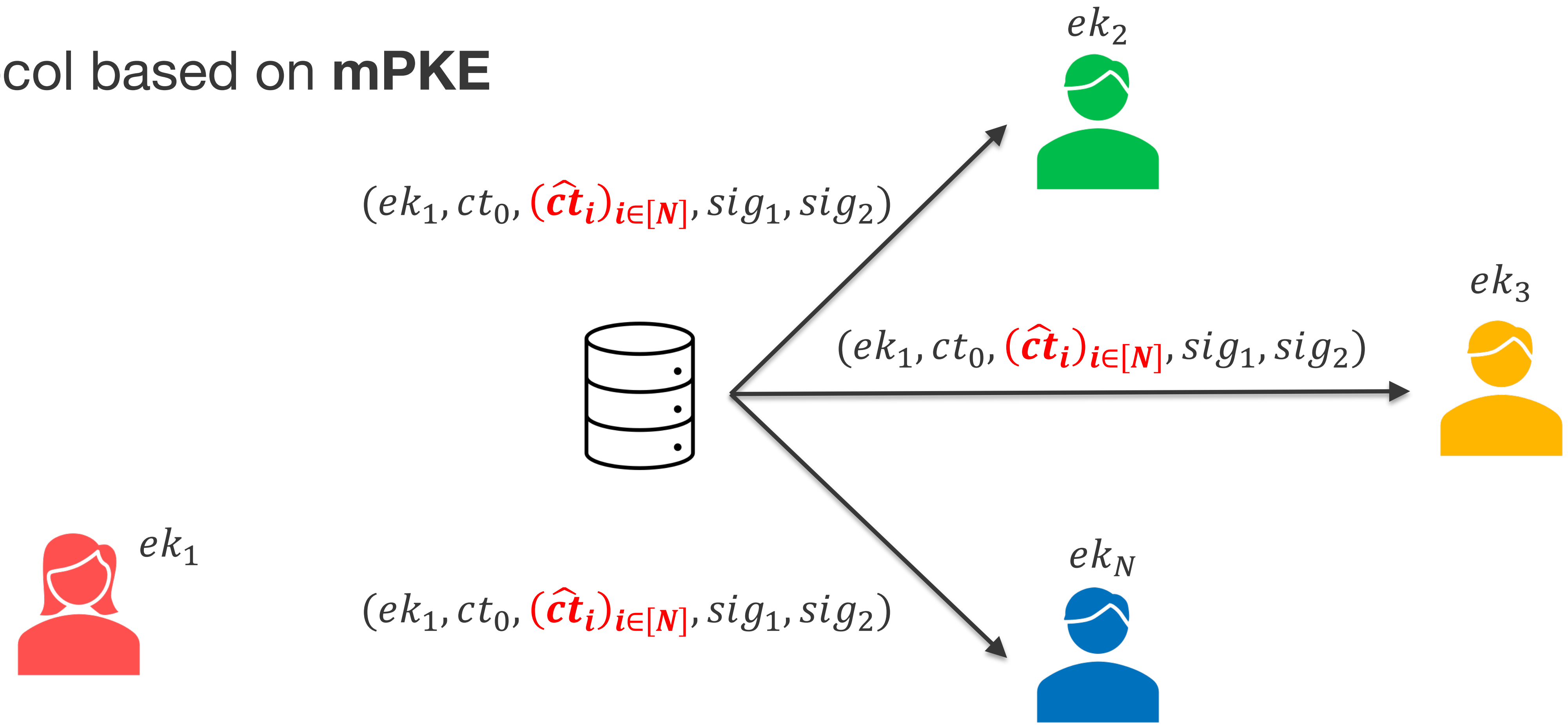
Chained mKEM [BBN19]

CGKA protocol based on **mPKE**



Drawback of Chained mKEM [BBN19]

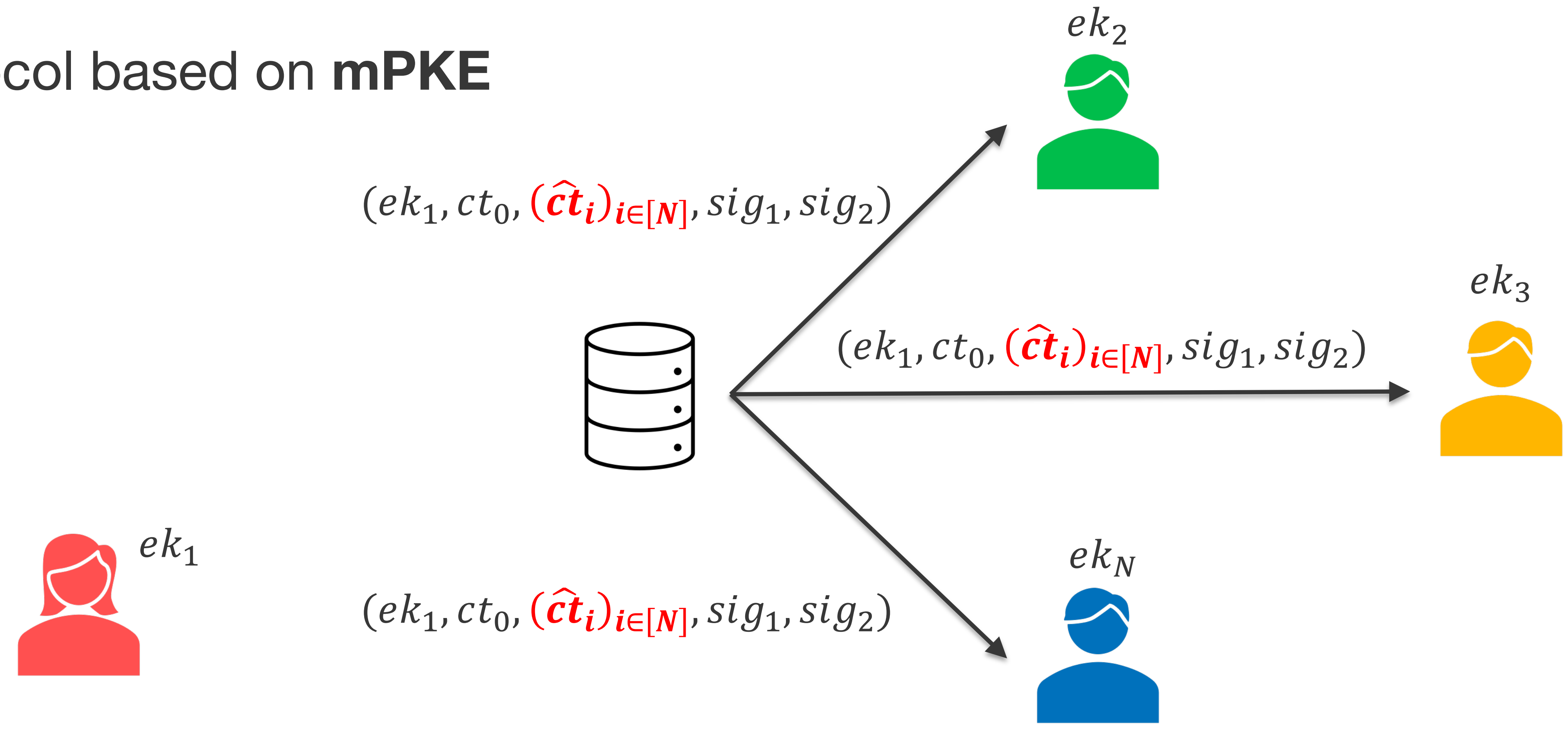
CGKA protocol based on **mPKE**



Download $O(N)$ ciphertexts to verify signature

Drawback of Chained mKEM [BBN19]

CGKA protocol based on **mPKE**

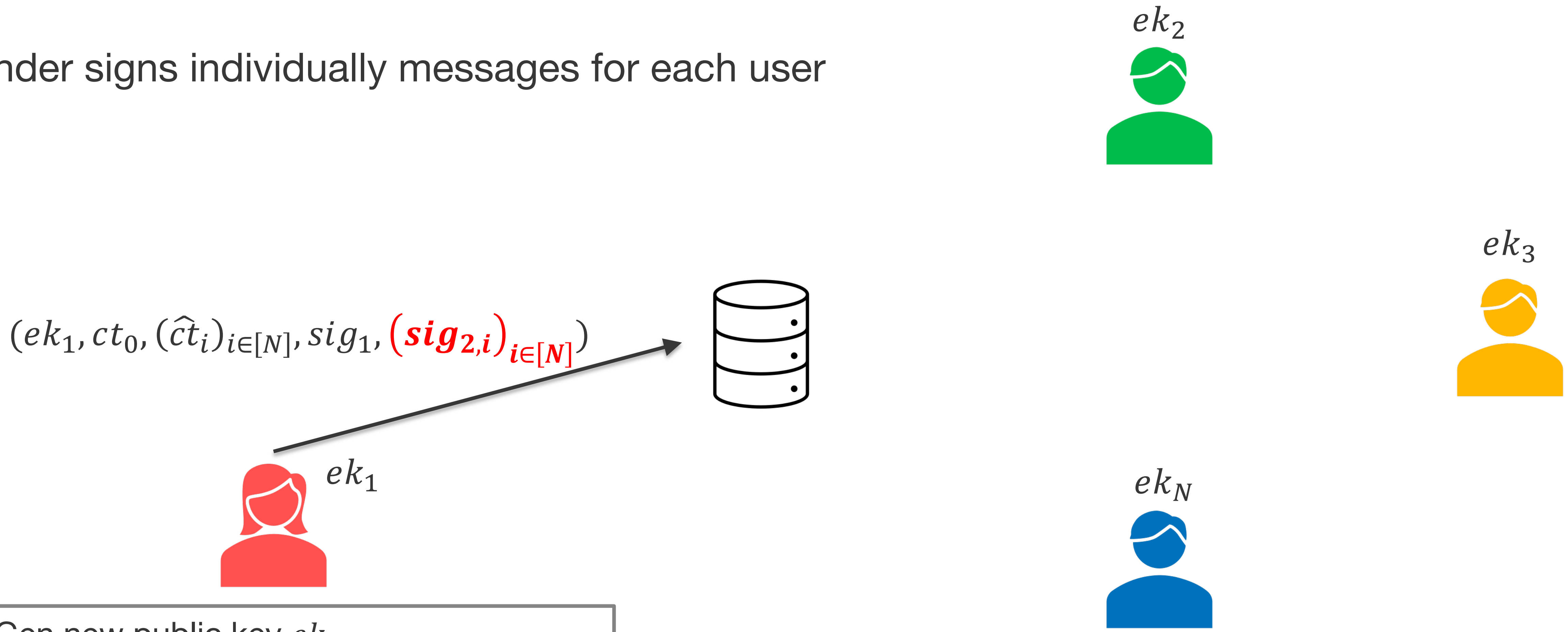


Can we reduce download cost to $O(1)$?

iphertexts
ature

Naïve approach to achieve $O(1)$ download cost

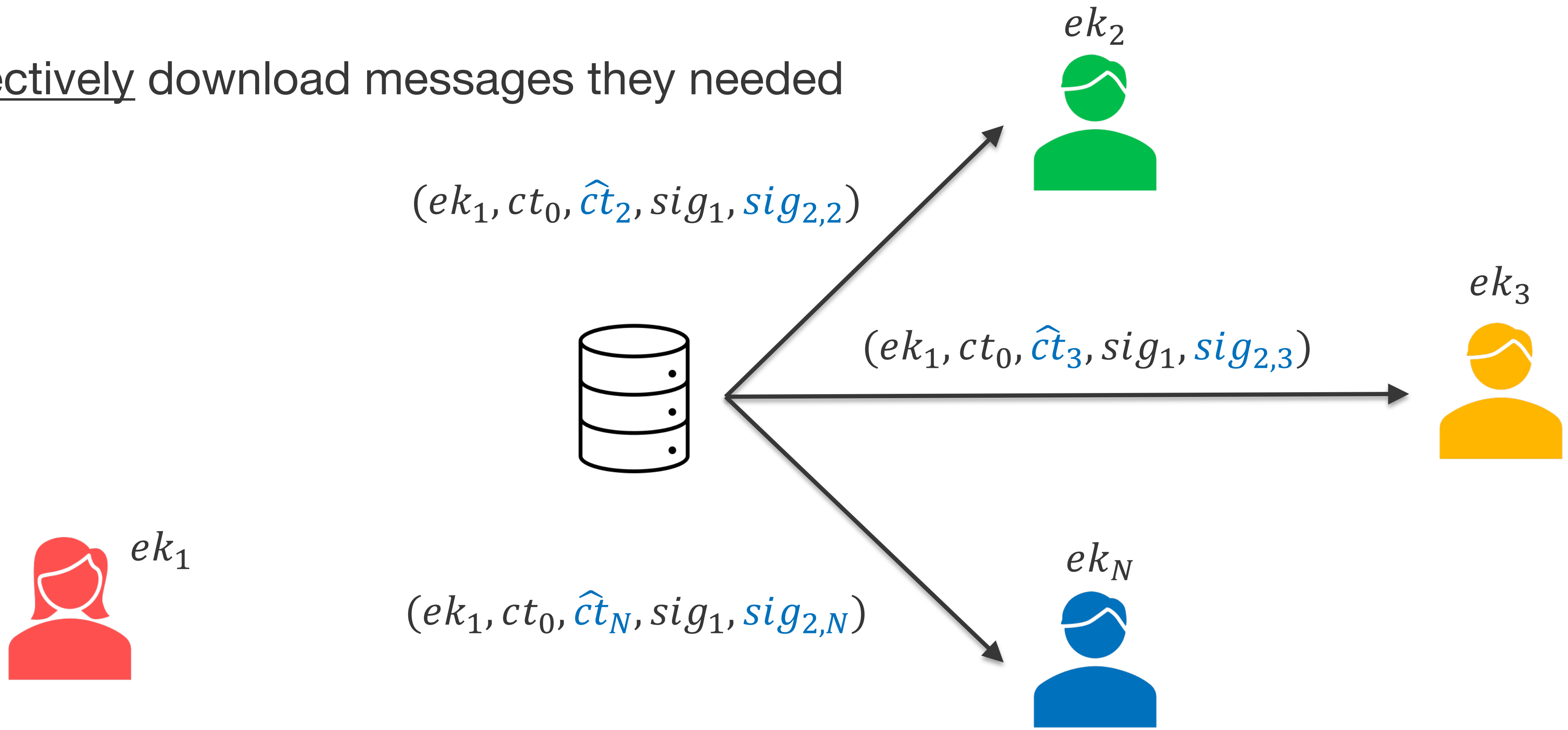
Sender signs individually messages for each user



1. Gen new public key ek_1
2. Gen new group key K
3. Gen $\left(ct_0, (\hat{ct}_i)_{i\in[N]}\right) \leftarrow \mathrm{mEnc}(K, (ek_i)_{i\in[N]})$
4. Gen $sig_1 \leftarrow \mathrm{Sign}(sk_1, ek_1)$
5. **For $\boldsymbol{i \in [N]}$, $\boldsymbol{sig_{2,i} \leftarrow \mathrm{Sign}(sk_1, (ct_0, \hat{ct}_i))}$**

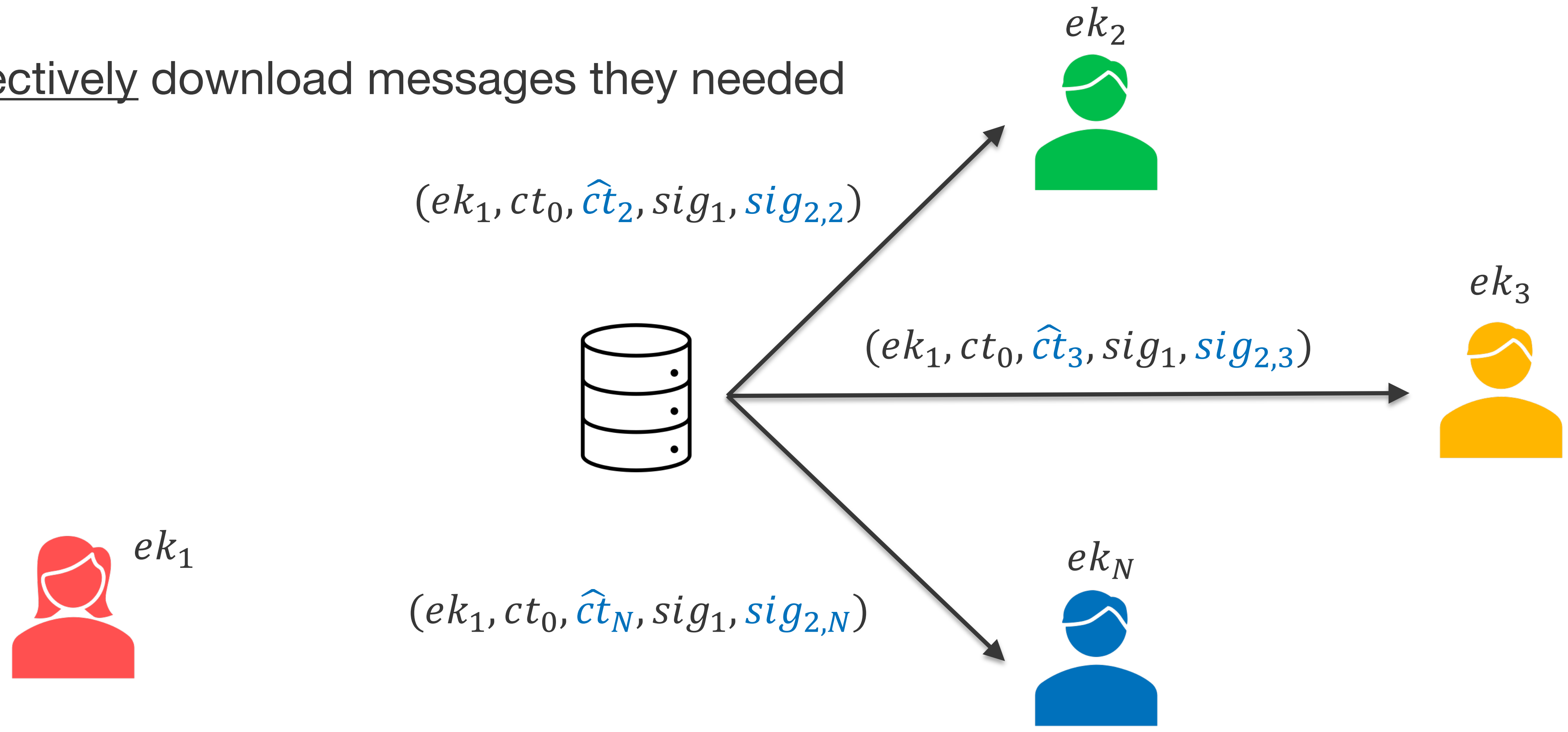
Naïve approach to achieve $O(1)$ download cost

Recipients selectively download messages they needed



Naïve approach to achieve $O(1)$ download cost

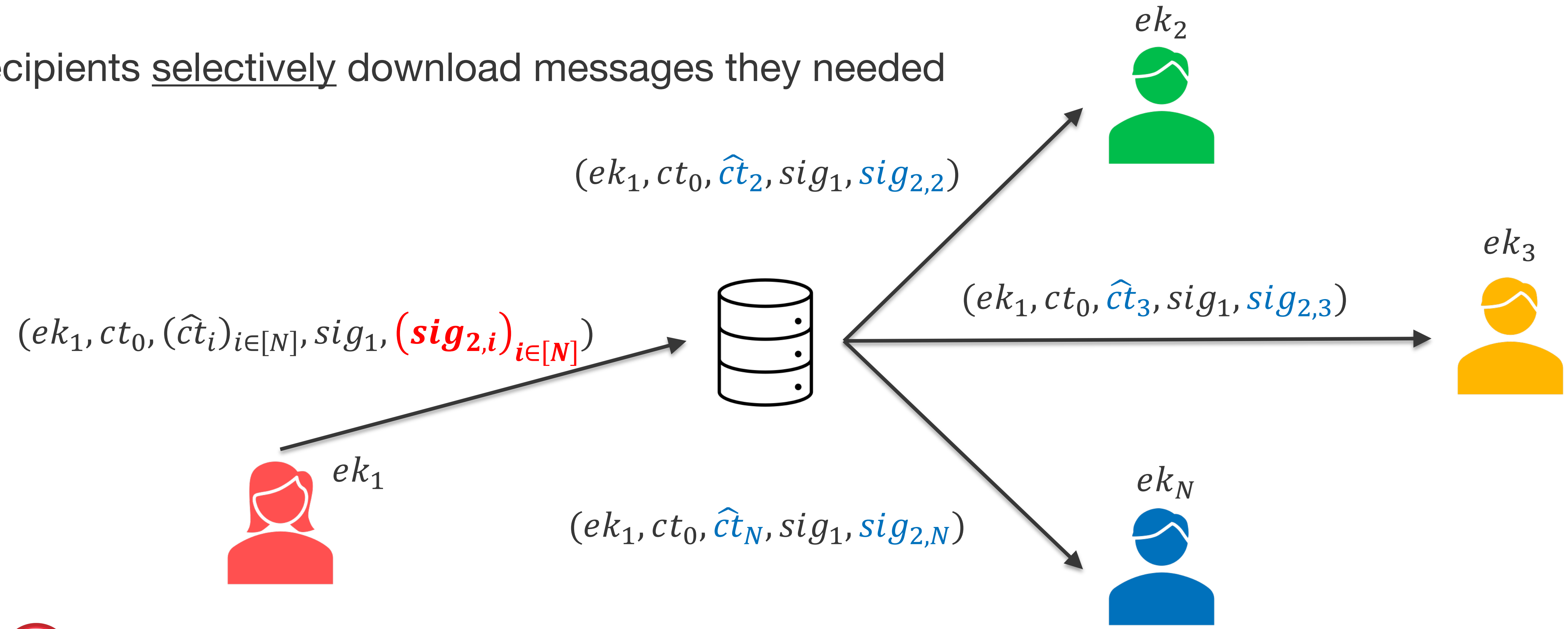
Recipients selectively download messages they needed



Download cost is $O(1)$

Naïve approach to achieve $O(1)$ download cost

Recipients selectively download messages they needed



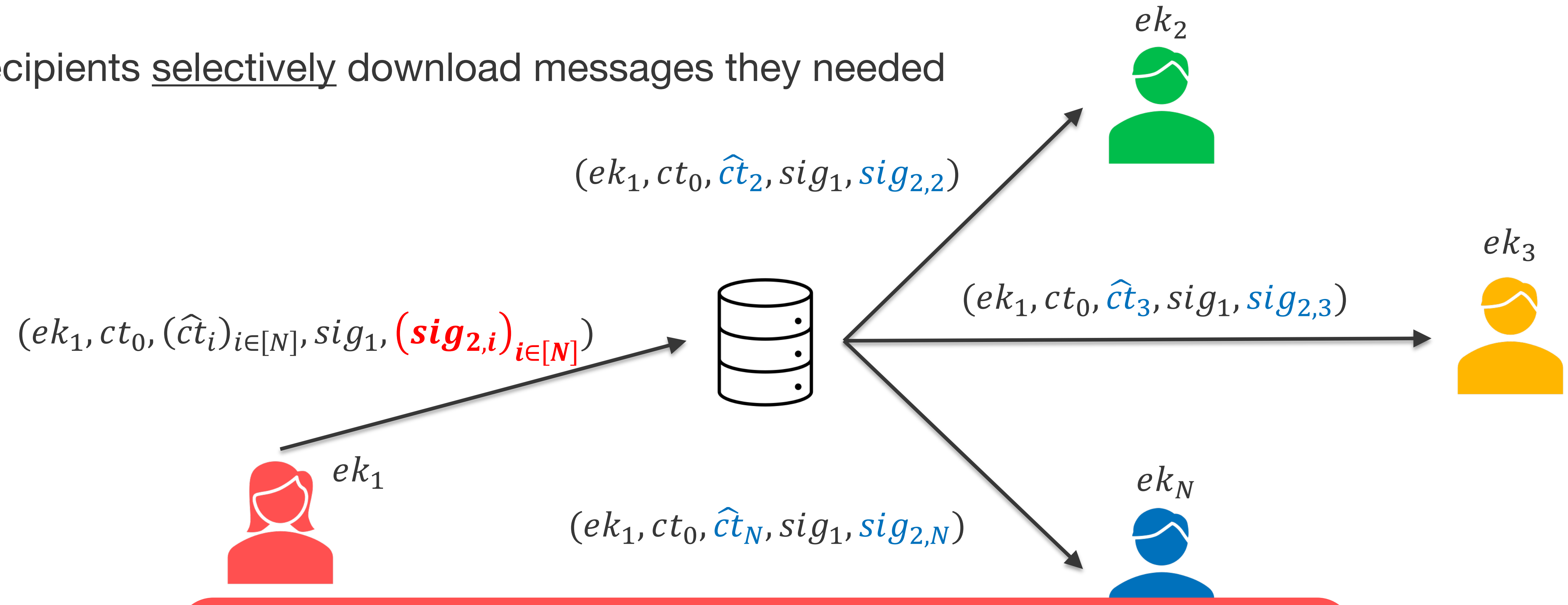
Upload $O(N)$ signatures

Note: PQ signatures are large!

Download cost is $O(1)$

Naïve approach to achieve $O(1)$ download cost

Recipients selectively download messages they needed



Upl

Note: PQ

$\boldsymbol{O(1)}$

Can we reduce download cost to $O(1)$
without increasing the num. of signatures?

Our solution: Committing mPKE (CmPKE)

ek_1

$$\mathrm{CmEnc}\big(M, (ek_i)_{i\in[N]}\big) \rightarrow (\mathrm{T}, (ct_i)_{i\in[N]})$$

ek_2

$(\mathrm{T}, ct_2) \rightarrow$

$$\mathrm{CmDec}\big(dk_2, (\mathrm{T}, ct_2)\big) \rightarrow M \text{ or } \perp$$

⋮

ek_N

$(\mathrm{T}, ct_N) \rightarrow$

$$\mathrm{CmDec}\big(dk_N, (\mathrm{T}, ct_N)\big) \rightarrow M \text{ or } \perp$$

Our solution: Committing mPKE (CmPKE)

ek_1

$\mathrm{CmEnc}\big(M, (ek_i)_{i\in[N]}\big) \to (\mathrm{T}, (ct_i)_{i\in[N]})$

ek_2

(T, ct_2) → 

$\mathrm{CmDec}\big(dk_2, (\mathrm{T}, ct_2)\big) \to M \text{ or } \bot$

⋮

ek_N

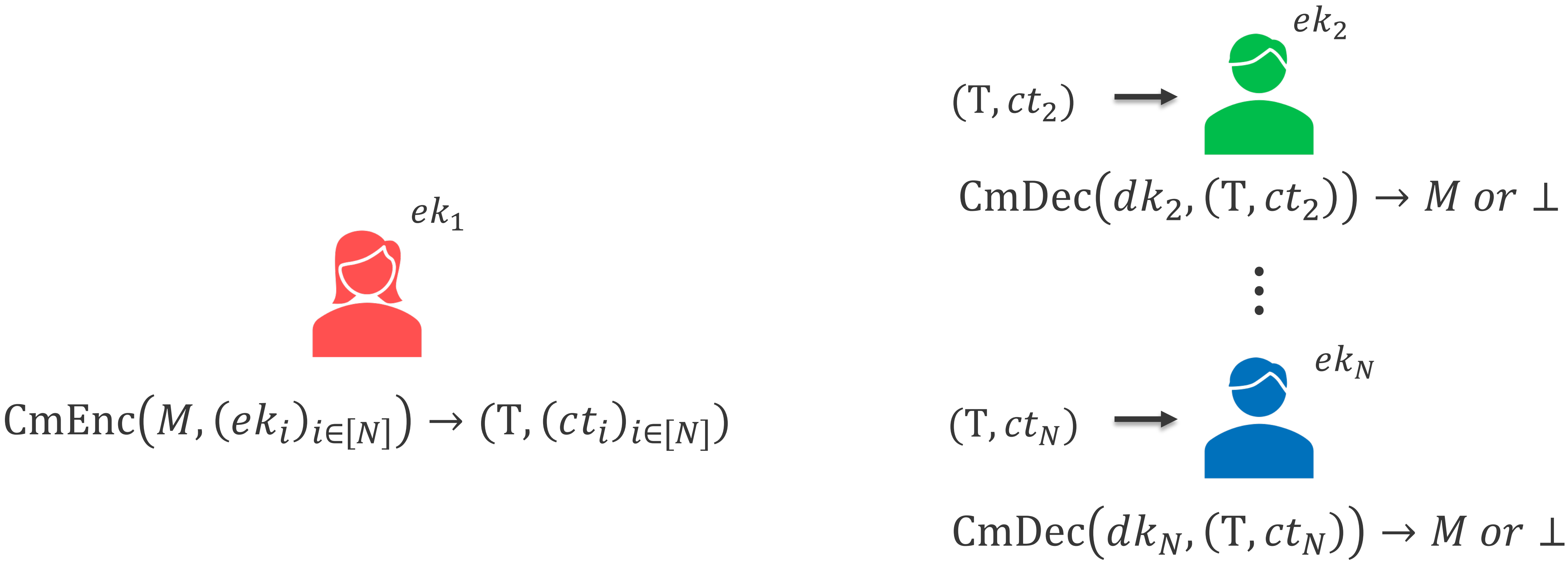
(T, ct_N) → 

$\mathrm{CmDec}\big(dk_N, (\mathrm{T}, ct_N)\big) \to M \text{ or } \bot$

Commitment-binding: T is linked to a unique message M

⇒ If parties receive the same T, they decrypt the same M or $\bot$

Our solution: Committing mPKE (CmPKE)



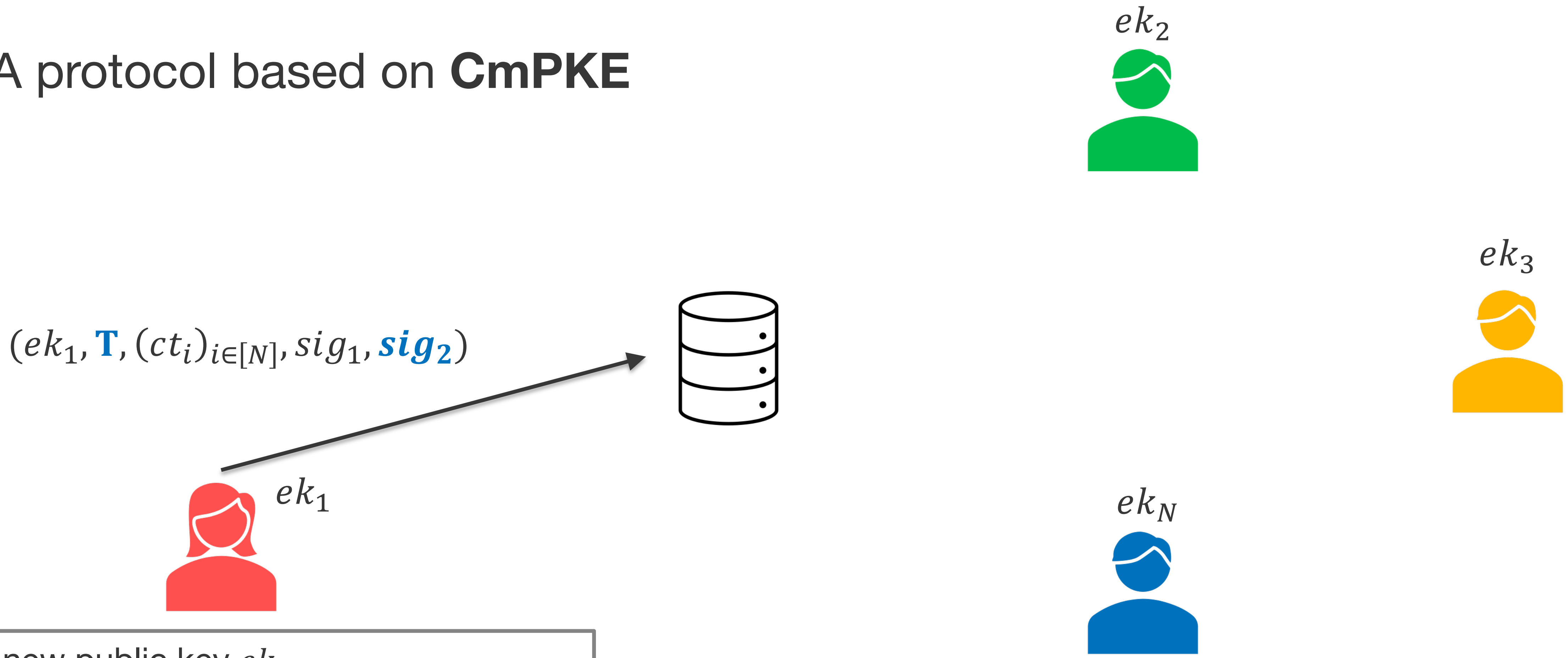
Propose IND-CPA mPKE ⇒ IND-CCA CmPKE transformation

- CmEnc runs $\mathrm{mEnc}\big(k, (ek_i)_{i \in [N]}\big) \rightarrow (\mathrm{ct}_0, (\widehat{ct}_i)_{i \in [N]})$ and $\mathrm{SKE.Enc}(k, M) \rightarrow c$
- Outputs $\mathrm{T} = (ct_0, c)$ and $ct_i = \widehat{ct}_i$, $|c| = 32$ bytes

Use key-committing AEADs [FOR17, GLR17, ADG+20] as SKE

Our CGKA: Chained CmPKE

CGKA protocol based on **CmPKE**

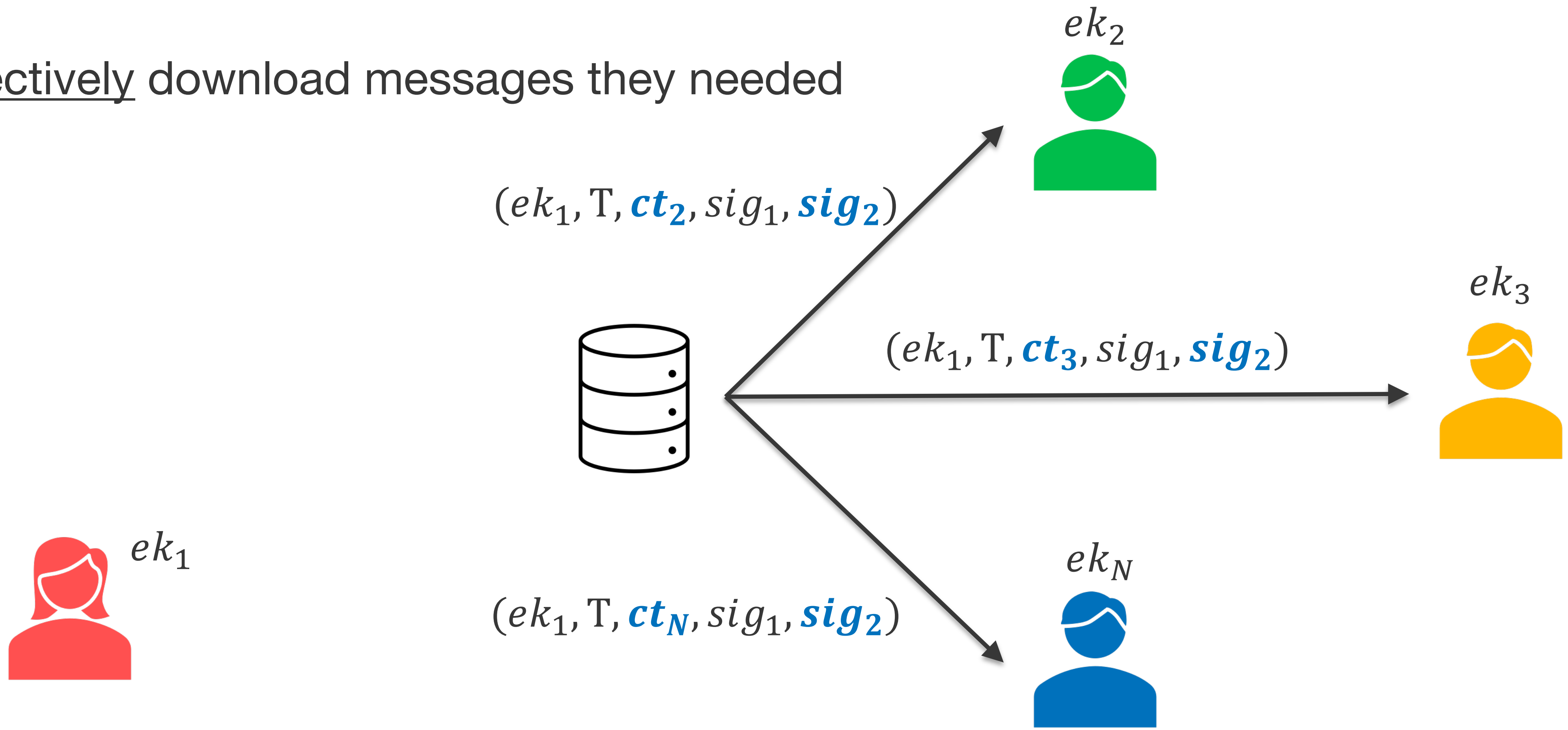


1. Gen new public key ek_1
2. Gen new group key K
3. **Gen $\left(\mathbf{T}, (\boldsymbol{ct_i})_{i\in[N]}\right) \leftarrow \mathbf{CmEnc}(\boldsymbol{K}, (\boldsymbol{ek_i})_{i\in[N]})$**
4. Gen $sig_1 \leftarrow \mathrm{Sign}(sk_1, ek_1)$
5. **Gen $\boldsymbol{sig_2} \leftarrow \mathbf{Sign}(\boldsymbol{sk_1}, \mathbf{T})$**

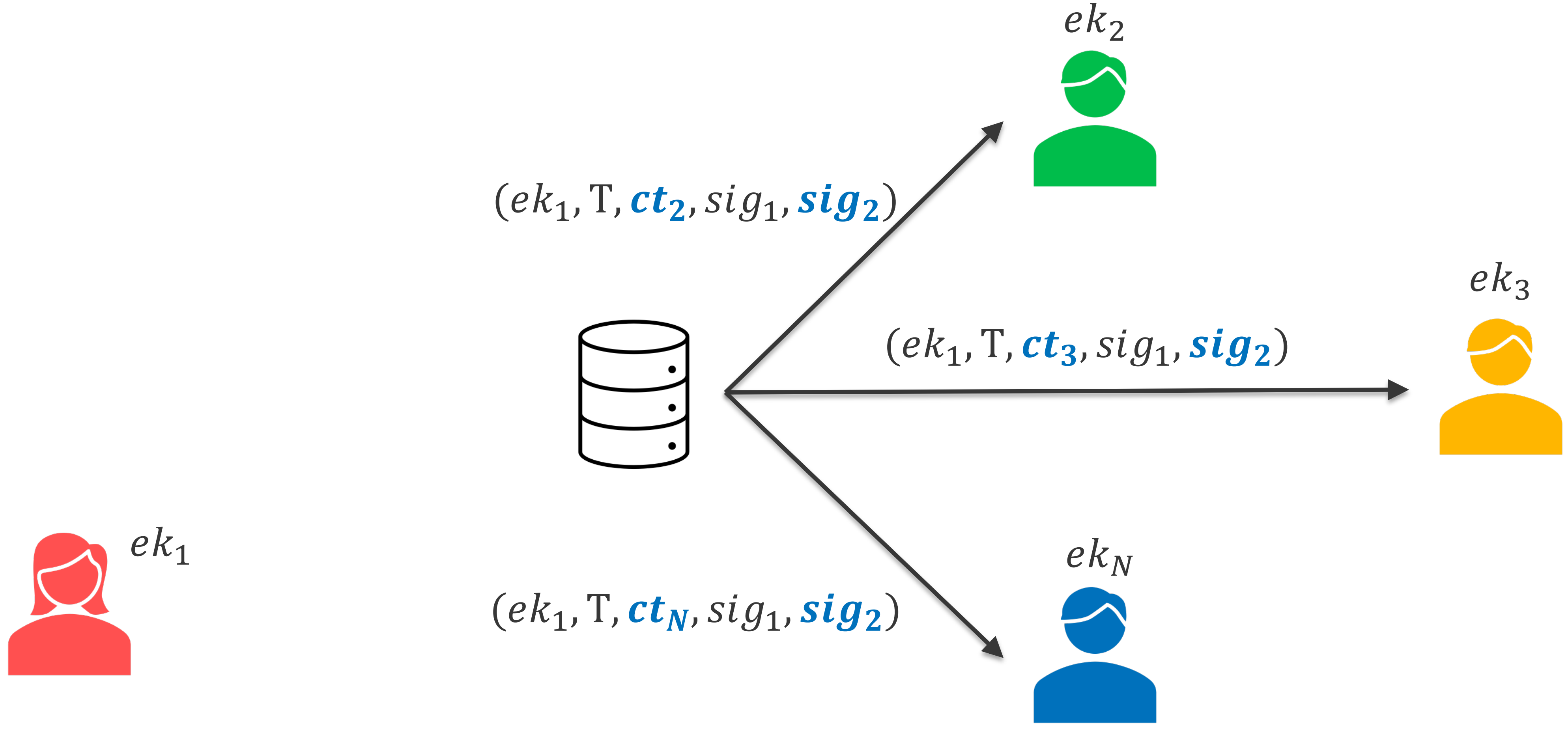
Due to commitment-binding, tampering with ct_i is detected at decryption

Our CGKA: Chained CmPKE

Recipients selectively download messages they needed

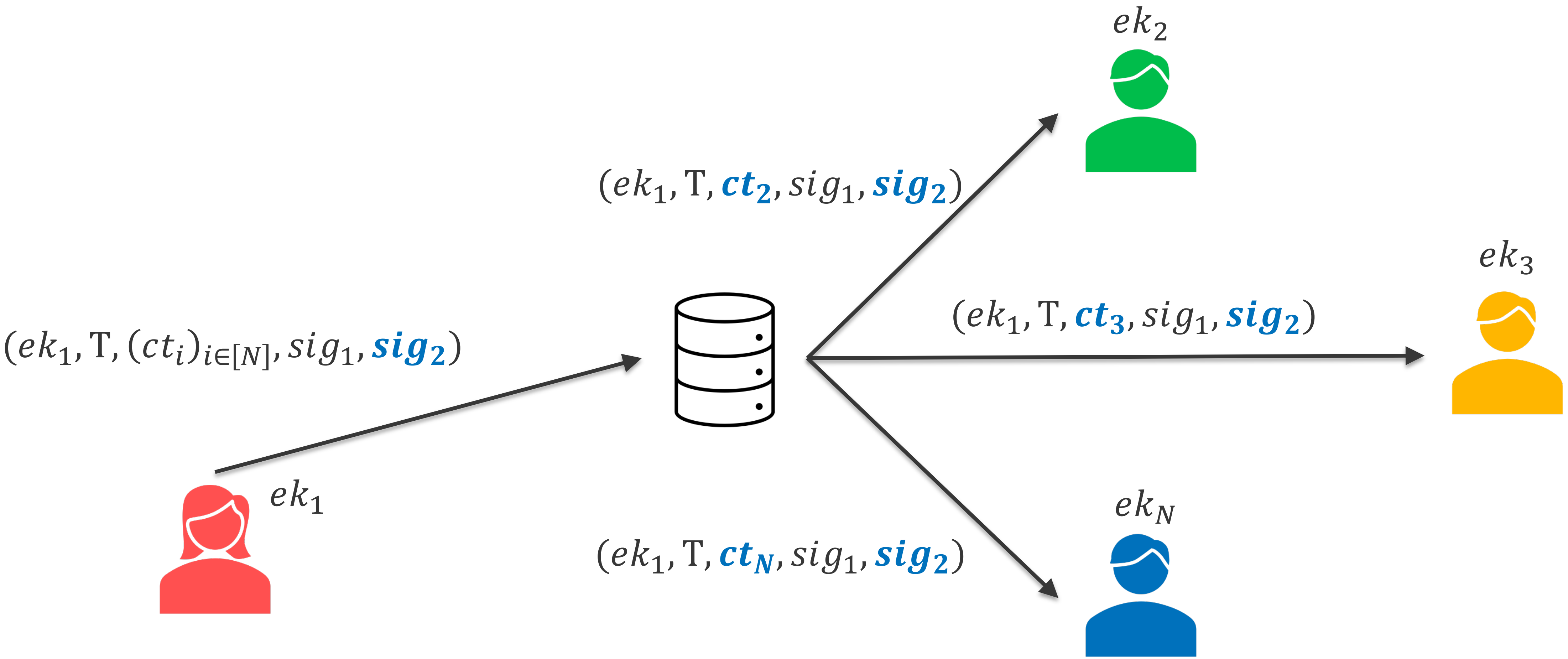


Our CGKA: Chained CmPKE



Download cost is $O(1)$

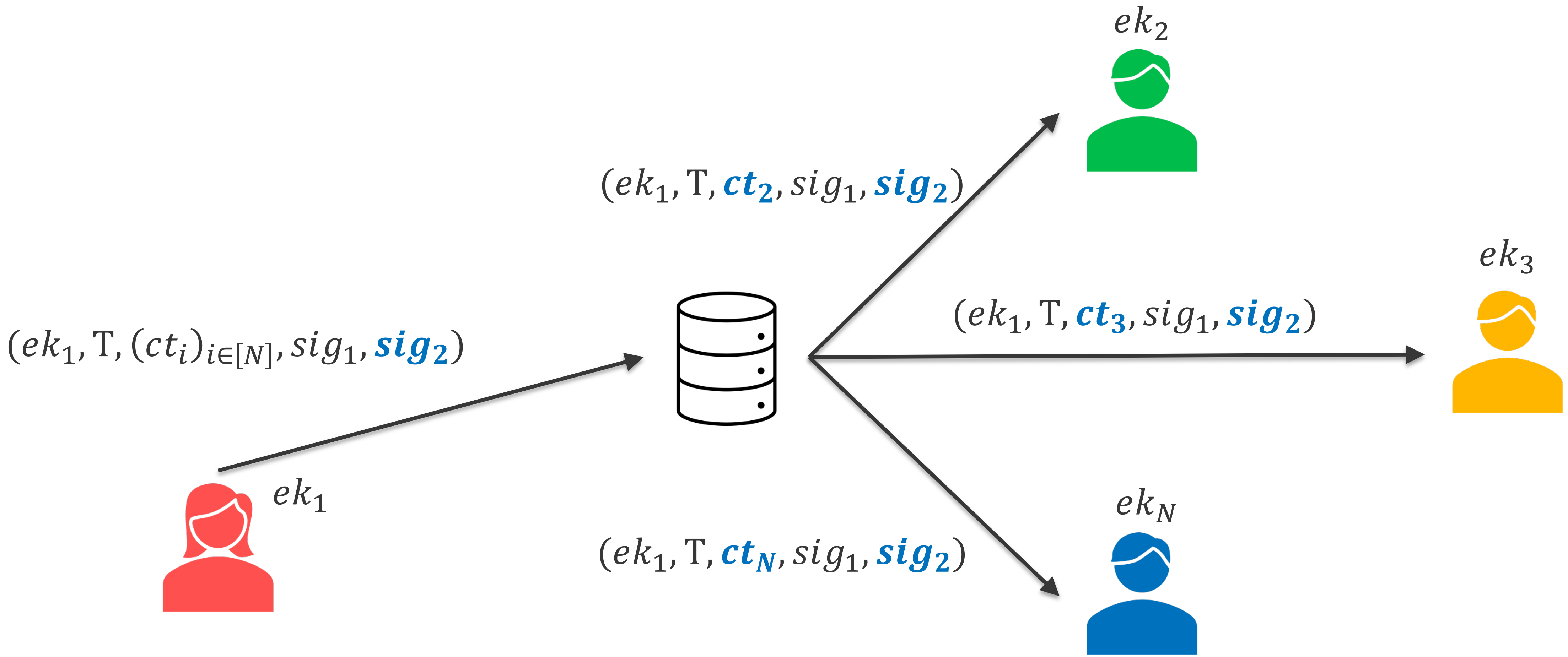
Our CGKA: Chained CmPKE



✅ **Upload constant signatures**

✅ **Download cost is $O(1)$**

Our CGKA: Chained CmPKE



Upl ... (1)

We achieve $O(1)$ download cost without increasing the num. of signatures! ☺

Security of Chained CmPKE

Chained CmPKE is as secure as TreeKEM version 10 in MLS

- Adopt the UC security model in [AJM20] used to analyze TreeKEM
  - It considers active adversaries and malicious insiders
- Extend this model to capture selective downloading of messages
  - Our model is the strict generalization of the model in [AJM20]

Contribution 2

More efficient post-quantum mPKEs

Existing post-quantum mPKE

[KKPP20] proposed efficient PQ mPKEs based on LWE, LWR, and SIDH.
Example scheme based on [LPR10, LP11]:

$\mathrm{Enc}(ek = \mathbf{B}, M)$:

1. Sample short matrixes $\mathbf{R}, \mathbf{E}', \mathbf{E}''$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. $\mathbf{V} \leftarrow \mathbf{RB} + \mathbf{E}'' + \mathrm{Encode}(M)$
4. $ct \coloneqq (\mathbf{U}, \mathbf{V})$

→

$\mathrm{mEnc}(\{ek_1, \dots, ek_N\}, M)$:

1. Sample short matrixes $\mathbf{R}, \mathbf{E}'$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. For $i = 1, \dots, N$
   1. Sample short matrix $\mathbf{E}_i''$
   2. $\mathbf{V}_i \leftarrow \mathbf{RB}_i + \mathbf{E}_i'' + \mathrm{Encode}(M)$
4. $(ct_0, (\widehat{ct}_i)_{i \in [N]}) \coloneqq (\mathbf{U}, (\mathbf{V}_i)_{i \in [N]})$

Existing post-quantum mPKE

[KKPP20] proposed efficient PQ mPKEs based on LWE, LWR, and SIDH.
Example scheme based on [LPR10, LP11]:

$\mathrm{Enc}(ek = \mathbf{B}, M)$:
1. Sample short matrixes $\mathbf{R}, \mathbf{E}', \mathbf{E}''$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. $\mathbf{V} \leftarrow \mathbf{RB} + \mathbf{E}'' + \mathrm{Encode}(M)$
4. $ct := (\mathbf{U}, \mathbf{V})$

→

$\mathrm{mEnc}(\{ek_1, \dots, ek_N\}, M)$:
1. Sample short matrixes $\mathbf{R}, \mathbf{E}'$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. For $i = 1, \dots, N$
    1. Sample short matrix $\mathbf{E}_i''$
    2. $\mathbf{V}_i \leftarrow \mathbf{RB}_i + \mathbf{E}_i'' + \mathrm{Encode}(M)$
4. $(ct_0, (\widehat{ct}_i)_{i \in [N]}) := (\mathbf{U}, (\mathbf{V}_i)_{i \in [N]})$

Two shortcomings of [KKPP20]:
1. Not optimize parameters to make $\widehat{ct}_i$ smaller
    - In CGKA setting, small $\widehat{ct}_i$ is desirable to reduce upload cost ($\sim |\widehat{ct}_i| \cdot N$)
2. Not analyze the hardness of underlying problems in mPKE setting

Existing post-quantum mPKE

[KKPP20] proposed efficient PQ mPKEs based on LWE, LWR, and SIDH.
Example scheme based on [LPR10, LP11]:

$\mathrm{Enc}(ek = \mathbf{B}, M)$:
1. Sample short matrixes $\mathbf{R}, \mathbf{E}', \mathbf{E}''$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. $\mathbf{V} \leftarrow \mathbf{RB} + \mathbf{E}'' + \mathrm{Encode}(M)$
4. $ct := (\mathbf{U}, \mathbf{V})$

→

$\mathrm{mEnc}(\{ek_1, \dots, ek_N\}, M)$:
1. Sample short matrixes $\mathbf{R}, \mathbf{E}'$
2. $\mathbf{U} \leftarrow \mathbf{RA} + \mathbf{E}'$
3. For $i = 1, \dots, N$
   1. Sample short matrix $\mathbf{E}_i''$
   2. $\mathbf{V}_i \leftarrow \mathbf{RB}_i + \mathbf{E}_i'' + \mathrm{Encode}(M)$
4. $(ct_0, (\widehat{ct_i})_{i\in[N]}) := (\mathbf{U}, (\mathbf{V}_i)_{i\in[N]})$

Two shortcomings of [KKPP20]:

1. No
   - …N)
2. No

We fix this two shortcomings ☺

Designing Lattice-Based mPKEs: Attacks and Toolkit

- **Attacks with $O(1)$ samples**
  - Lattice (primal)
  - Lattice (dual)
  - Decoding

- **Attacks with many samples**
  - Arora-Ge: requires $n^{O(d)}$ samples (d = cardinality of error support)
  - BKW

- **Toolkit**
  - *Bit dropping*
    + \+ Decrease $|\mathbf{V}_i|$
    + \+ Increase the LWE noise
    + – Increase decryption failure
  - *Coefficient dropping*
    + \+ Decrease $|\mathbf{V}_i|$
  - *Increase the modulus q*
    + \+ Pack more bits / coefficient
    + – Increase $|\mathbf{U}_i|$
    + – Decrease the LWE noise

Designing Lattice-Based mPKEs: Attacks and Toolkit

- **Attacks with $O(1)$ samples**
  - Lattice (primal)
  - Lattice (dual)
  - Decoding

Good for security!

- **Attacks with many samples**
  - Arora-Ge: requires $n^{O(d)}$ samples (d = cardinality of error support)
  - BKW

- **Toolkit**
  - *Bit dropping*
    - \+ Decrease $|\mathbf{V}_i|$
    - \+ Increase the LWE noise
    - – Increase decryption failure

Good for efficiency!

  - *Coefficient dropping*
    - \+ Decrease $|\mathbf{V}_i|$
  - *Increase the modulus q*
    - \+ Pack more bits / coefficient
    - – Increase $|\mathbf{U}_i|$
    - – Decrease the LWE noise

Comparison: new parameters vs. existing parameters

Bandwidth of mPKE based on existing parameters (blue) and new parameters (blank)

Size in byte. Security level is NIST I (≧ AES-128).

mPKE scheme	$\lvert ek \rvert$	$\lvert ct_0 \rvert$	$\lvert \widehat{ct_i} \rvert$
Kyber512 [SAB+ 20]	768(+32)	640	128
Ilum512	768	704	**48**
LPRime653 [BBC+ 20]	865(+32)	865(+32)	128
LPRime757	1076	1076	**32**
Frodo640 [NAB+ 20]	9600(+16)	9600	120
Bilbo640	10240	10240	**24**
SIKEp434 [JAC+ 20]	330	330	16

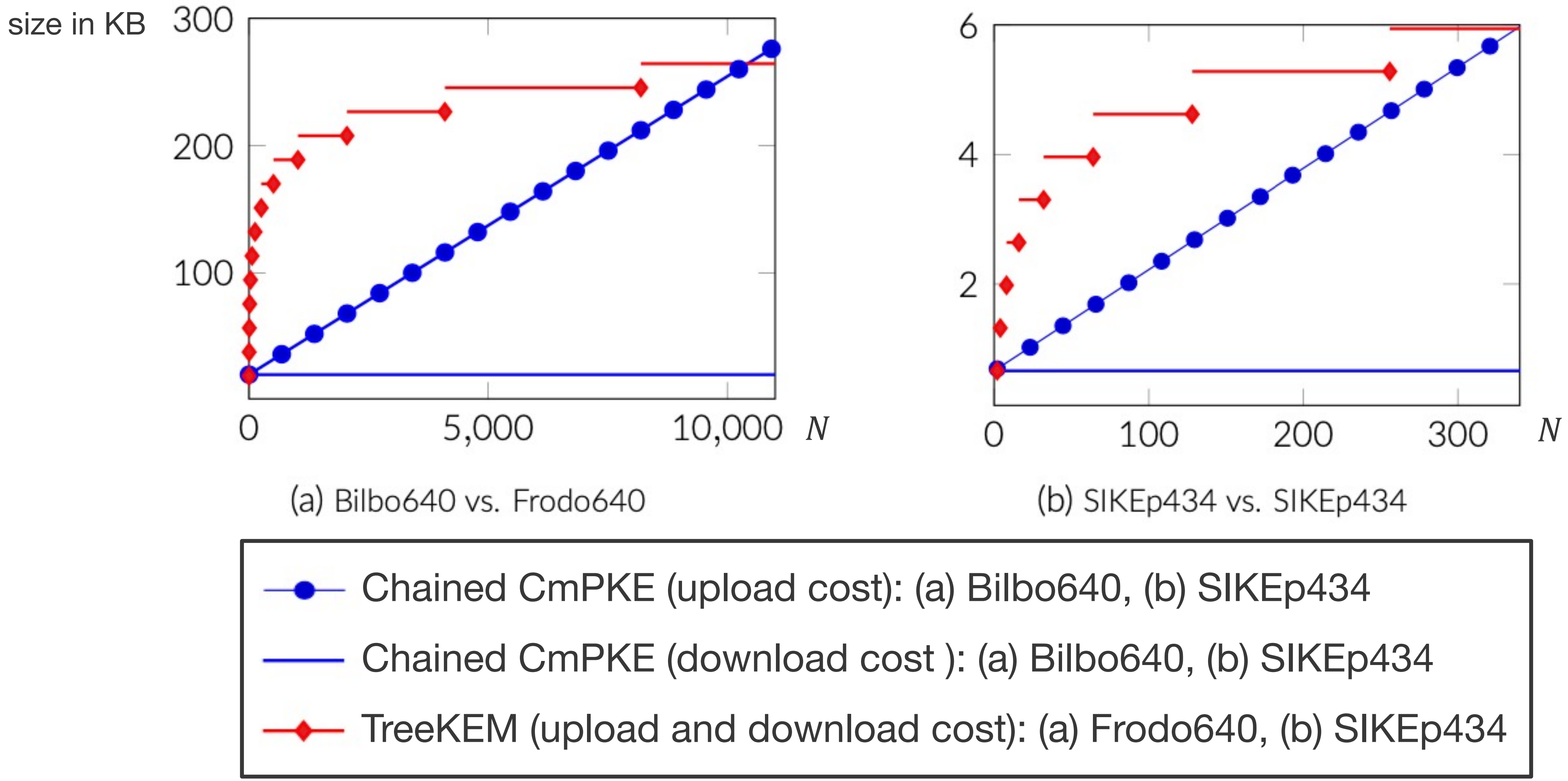
$\lvert \widehat{ct_i} \rvert$ is reduced by 60-80% at the cost of slightly increase in $\lvert ek \rvert$ and $\lvert ct_0 \rvert$

⇒ Minimize the concrete size of key update messages ($\sim \lvert \widehat{ct_i} \rvert \cdot N$)

Comparison and Implementation

Chained CmPKE vs. TreeKEM: upload and download cost

Size of key update messages in Kilobyte (y-axis) depending on the group size (x-axis)

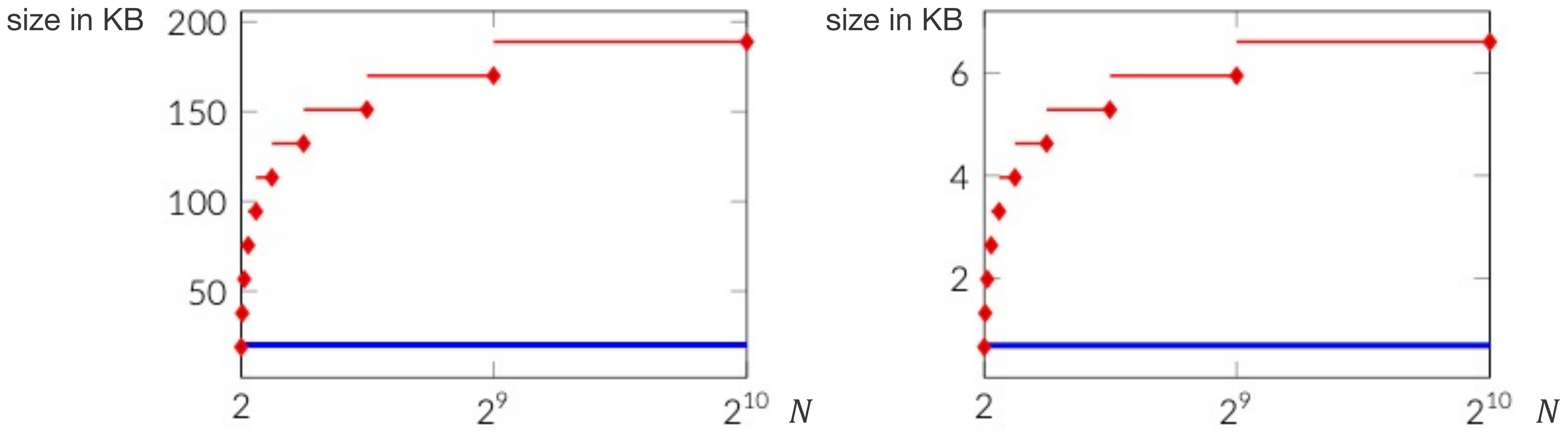


(a) Bilbo640 vs. Frodo640

(b) SIKEp434 vs. SIKEp434

Chained CmPKE vs. TreeKEM: total cost (normalized by N)

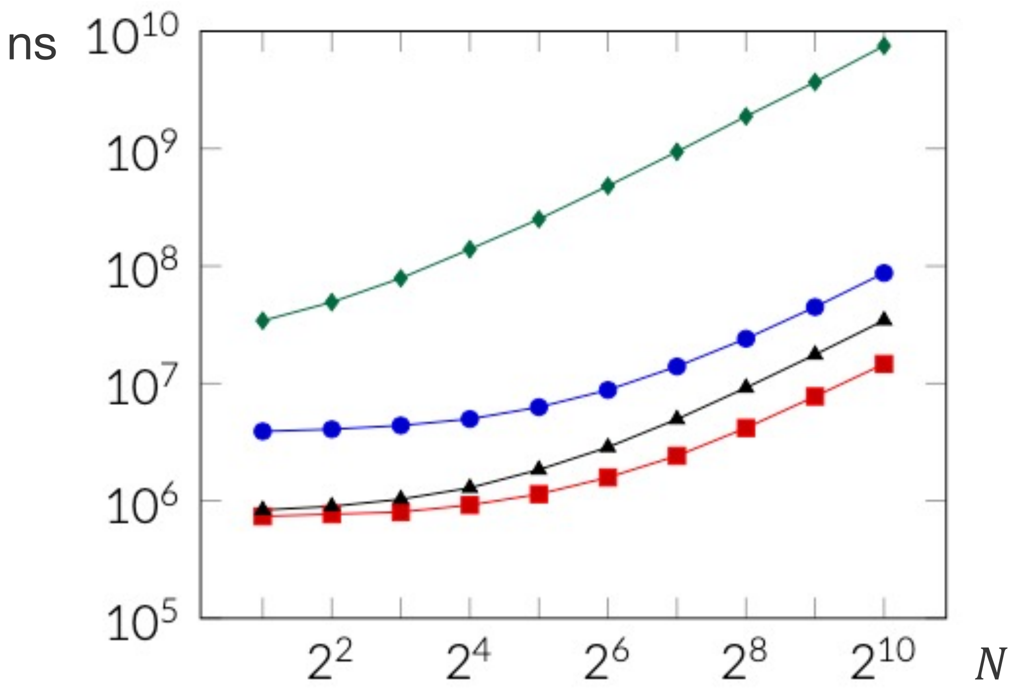
Total cost of key update in Kilobyte (y-axis) depending on the group size N (x-axis)



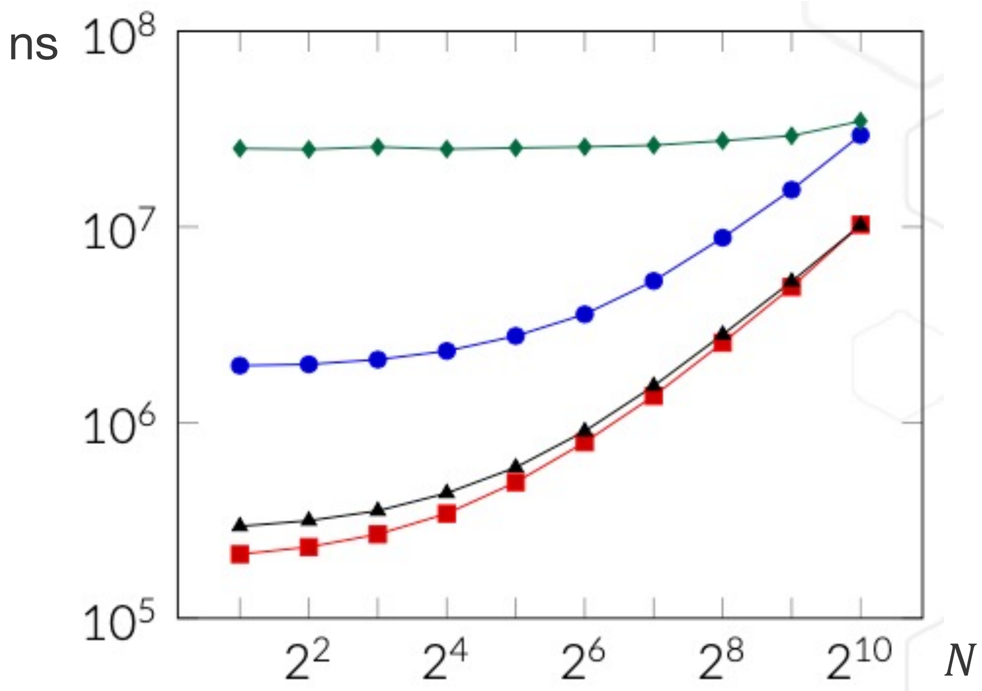
(a) Bilbo640 vs. Frodo640

(b) SIKEp434 vs. SIKEp434

Chained CmPKE: computation cost



(a) Generate key update messages



(b) Process received messages

Execution time in nanoseconds of some procedures as a function of group size for Ilum512 (red square), LPRime757 (black triangle), Bilbo640 (blue circle), SIKEp434 (green diamond).
Log-scale. Times are obtained on Apple M1@3.2 GHz.

Conclusion

Chained CmPKE: CGKA with asymmetric bandwidth cost

Scheme	Upload cost	Download cost	Total cost (upload + N-1 download)
TreeKEM	$\Omega(\log N)$	$\Omega(\log N)$	$\Omega(N \log N)$
Chained mKEM	$O(N)$	$O(N)$	$O(N^2)$
Chained CmPKE	$O(N)^\star$	$\boldsymbol{O(1)}$	$\boldsymbol{O(N)}$

$\star$: When N is about hundreds, the concrete upload cost is smaller than TreeKEM.

Chained CmPKE is based on Chained mKEM with two new ideas:

1. **Committing mPKE** ⇒ achieve $O(1)$ download cost
2. **More efficient PQ mPKE** ⇒ reduce the size of key update messages

References

- [CGCD+17] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila. A Formal Security Analysis of the Signal Messaging Protocol. EuroS&P 2017.
- [ACD19] J. Alwen, S. Coretti, and Y. Dodis. The double ratchet: Security notions, proofs, and modularization for the signal protocol. EUROCRYPT 2019.
- [BFG+20] J. Brendel, M. Fischlin, F. Günther, C. Janson, and D. Stebila, "Towards Post-Quantum Security for Signal's X3DH Handshake. SAC 2020.
- [HKKP21] K. Hashimoto, S. Katsumata, K. Kwiatkowski, and T. Prest. An Efficient and Generic Construction for Signal's Handshake (X3DH): Post-Quantum, State Leakage Secure, and Deniable. PKC 2021.
- [ACDT20] J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis. Security analysis and improvements for the ietf mls standard for group messaging. CRYPTO 2020.
- [BBN18] K. Bhargavan, R. Barnes, and E. Rescorla. TreeKEM : Asynchronous Decentralized Key Management for Large Dynamic Groups. *A protocol proposal for Messaging Layer Security (MLS)*. 2018.
- [BBN19] K. Bhargavan, B. Beurdouche, and P. Naldurg. Formal Models and Verified Protocols for Group Messaging: Attacks and Proofs for IETF MLS. Research Report. 2019.
- [ACJM20] J. Alwen, S. Coretti, D. Jost, and M. Mularczyk. Continuous Group Key Agreement with Active Security. TCC 2020.
- [AJM20] J. Alwen, D. Jost, and M. Mularczyk. On The Insider Security of MLS. IACR ePrint. 2020.

References

- [OBR+21] E. Omara, B. Beurdouche, E. Rescorla, S. Inguva, A. Kwon, and A. Duric. The Messaging Layer Security MLS Architecture - draft-ietf-mls-architecture-06. *Internet Engineering Task Force Draft*. 2021.
- [BBM+20] R. Barnes, B. Beurdouche, J. Millican, E. Omara, K. Cohn-Gordon, and R. Robert. The Messaging Layer Security - draft-ietf-mls-protocol-11. *Internet Engineering Task Force Draft*. 2020.
- [FOR17] P. Farshim, C. Orlandi, and R. Rosie. Security of symmetric primitives under incorrect usage of keys. IACR Transactions on Symmetric Cryptology, pages 449–473, 2017.
- [GLR17] P. Grubbs, J. Lu, and T. Ristenpart. Message franking via committing authenticated encryption. CRYPTO 2017.
- [ADG+20] A. Albertini, T. Duong, S. Gueron, S. Kölbl, A. Luykx, and S. Schmieg. How to abuse and fix authenticated encryption without key commitment. To appear in USENIX Security 2022.
- [KKPP20] S. Katsumata, K. Kwiatkowski, F. Pintore, and T. Prest. Scalable Ciphertext Compression Techniques for Post-quantum KEMs and Their Applications. ASIACRYPT 2020.
- [LPR10] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. EUROCRYPT 2010.
- [LP11] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. CT-RSA 2011.

References

- [ABC+20] M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, K. G. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C. Jung Tjhai, M. Tomlinson, and W. Wang. Classic McEliece. Technical report, National Institute of Standards and Technology, 2020.
- [SAB+20] P. Schwabe, R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, G. Seiler, and D. Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020.
- [BBC+20] D. J. Bernstein, B. Bob Brumley, M. Chen, C. Chuengsatiansup, T. Lange, A. Marotzke, B. Peng, N. Tuveri, C. van Vredendaal, and B. Yang. NTRU Prime. Technical report, National Institute of Standards and Technology, 2020.
- [NAB+20] M. Naehrig, E. Alkim, J. Bos, L. Ducas, K. Easterbrook, B. LaMacchia, P. Longa, I. Mironov, V. Nikolaenko, C. Peikert, A. Raghunathan, and . Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2020.
- [JAC20+] D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, L. De Feo, B. Hess, A. Jalali, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, J. Renes, V. Soukharev, D. Urbanik, G. Pereira, K. Karabina, and A. Hutchinson. SIKE. Technical report, National Institute of Standards and Technology, 2020.