ACM CCS 2025 – Taipei, Taïwan

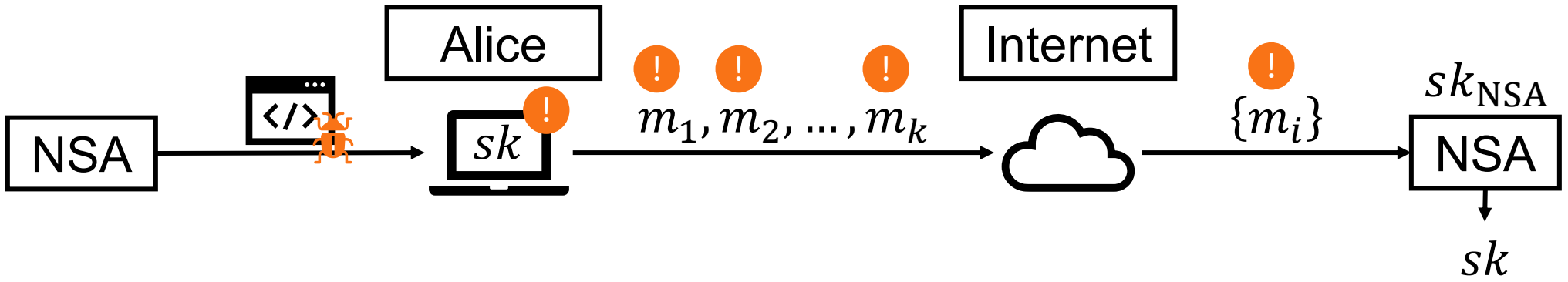
Subversion-resilient Key-exchange in the Post-quantum World (PQ-SRAKE)

Kevin DUVERGER, Pierre-Alain FOUQUE, Charlie JACOMME,
Guilhem NIOT, Cristina ONETE

10 / 14 / 2025

Introduction

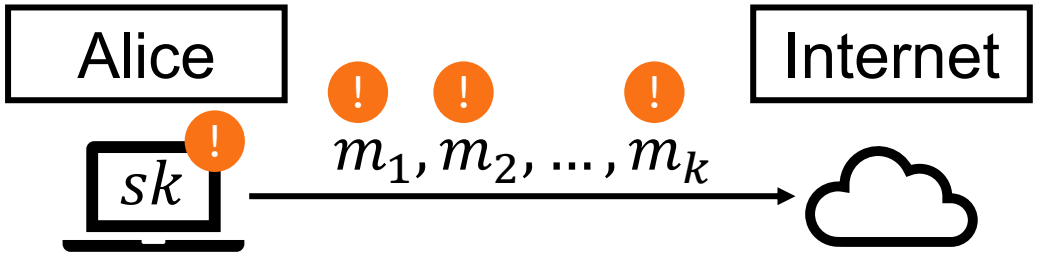
Introduction – Subversions

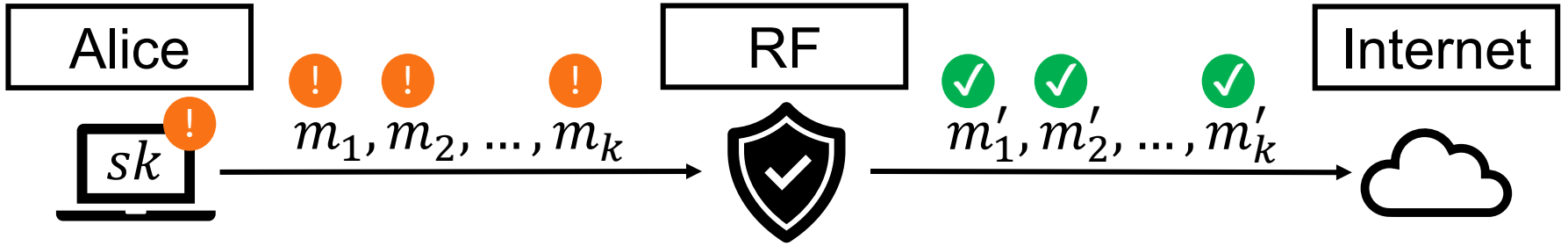


REAL ATTACKS ! Dual EC (2013), XZ Utils (2024), …

Introduction – Reverse firewalls

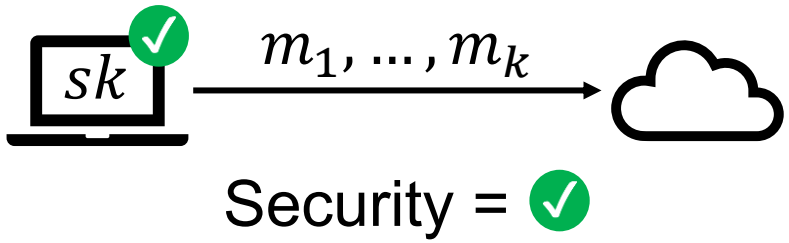
Ilya Mironov and Noah Stephens-Davidowitz - 2014 :
Reverse firewalls (RFs)

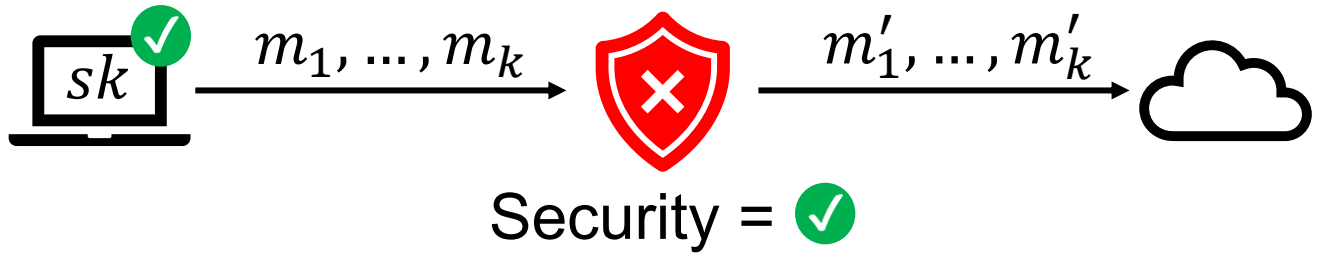


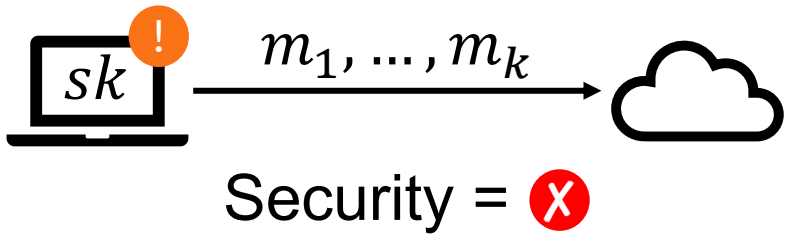


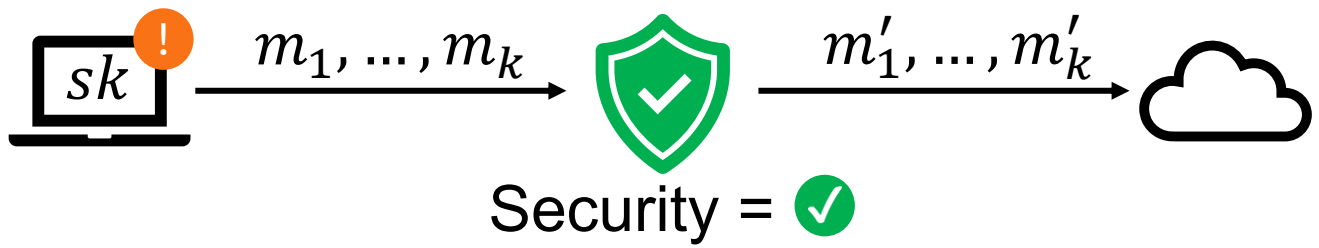
$$m'_i = Rerandomize(m_i, r)$$

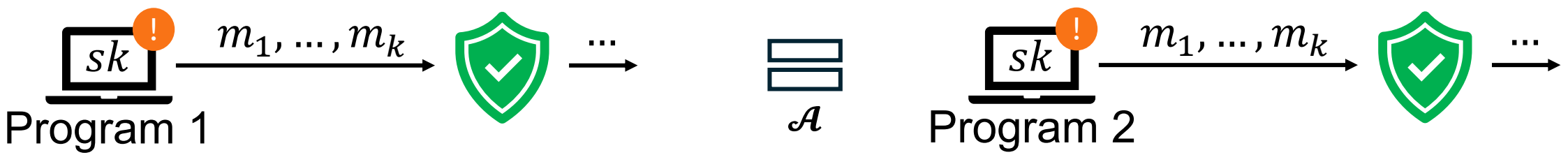
Introduction – Reverse firewalls











Introduction – Our goals

- Bossuat et al. 2020. Designing Reverse Firewalls for the Real World
  Signed DH secure (client-side) against subversion attacks

Questions

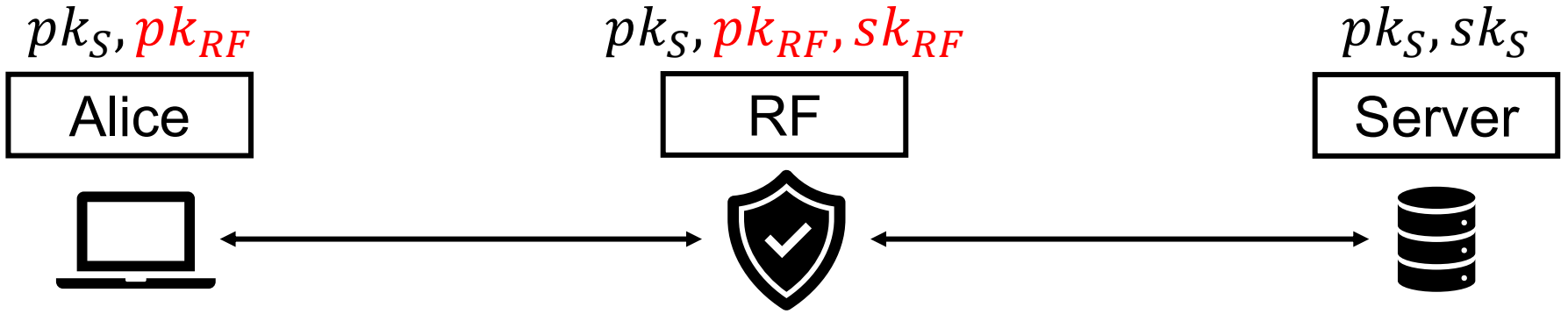
1. Efficiency + deployability
2. PQ-TLS RF
3. New primitives
4. Formal verification

Final goal : a protocol such that a router / vpn (that we don't trust) can heal the key-exchange

Real world RFs

Real world RFs – Setup-freeness

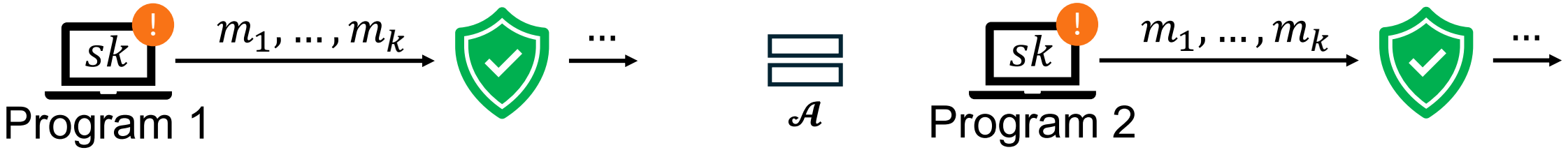
Bossuat et al. 2020 – Designing Reverse Firewalls for the Real World



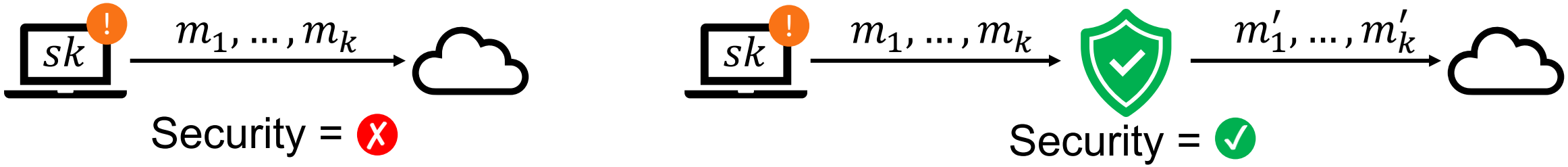
Shared parameters: complicated setup !

Setup-free RF – A RF that shares no long-term parameters with its party

Real world RFs – Relaxing security notions



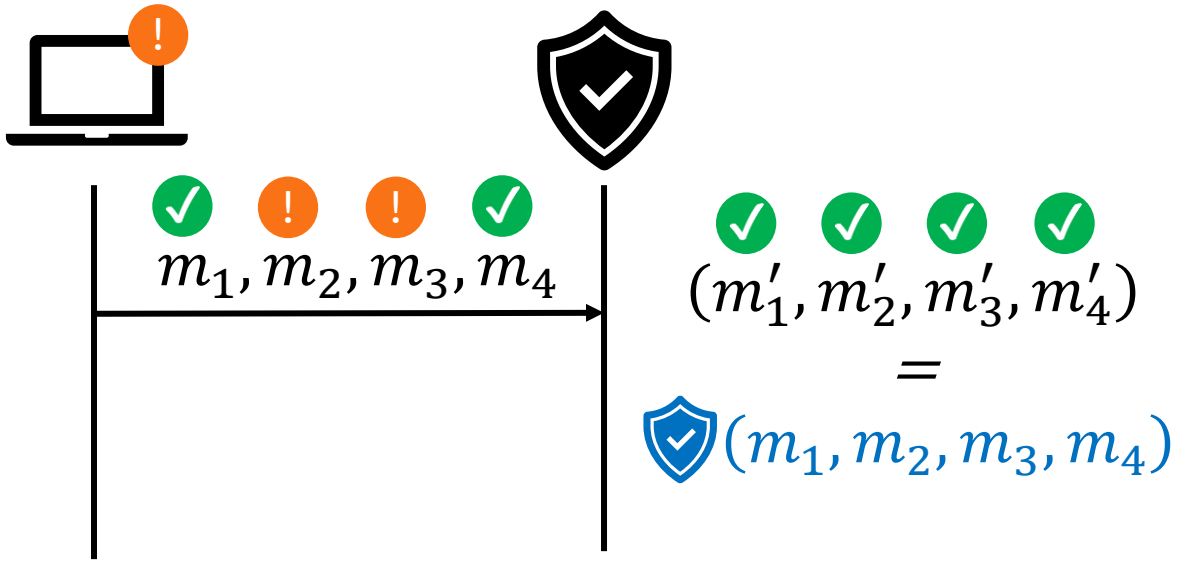
Exfiltration-resilience is **too strong** !



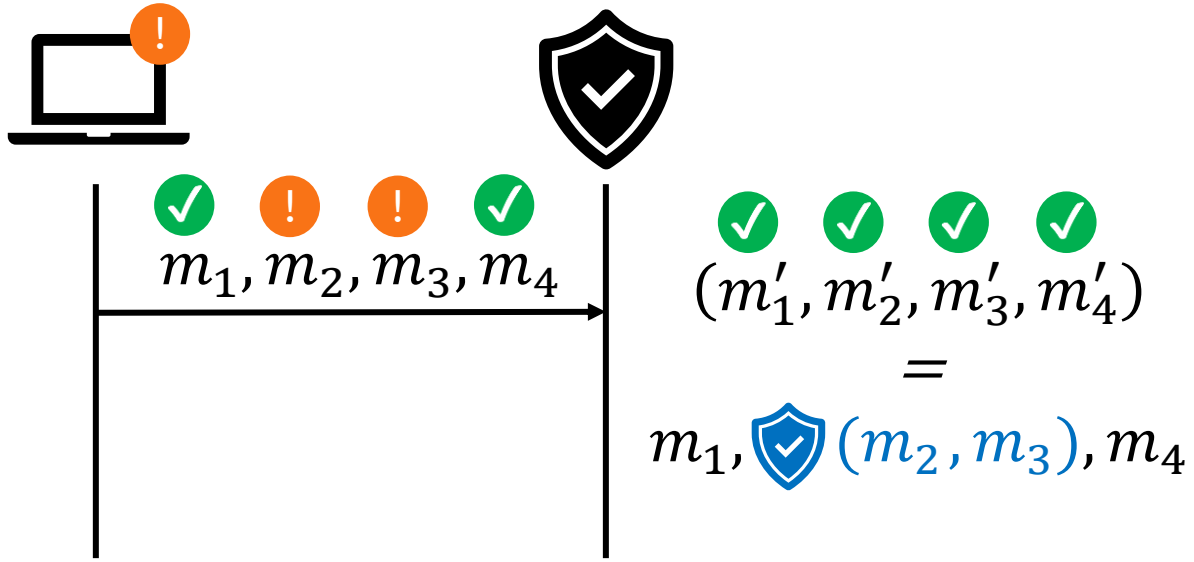
Security preservation = **main target** !

Real world RFs – Compromise scenario

All algorithms subverted: **too strong** in some cases
Might have trust in some algorithms !



Inefficient !

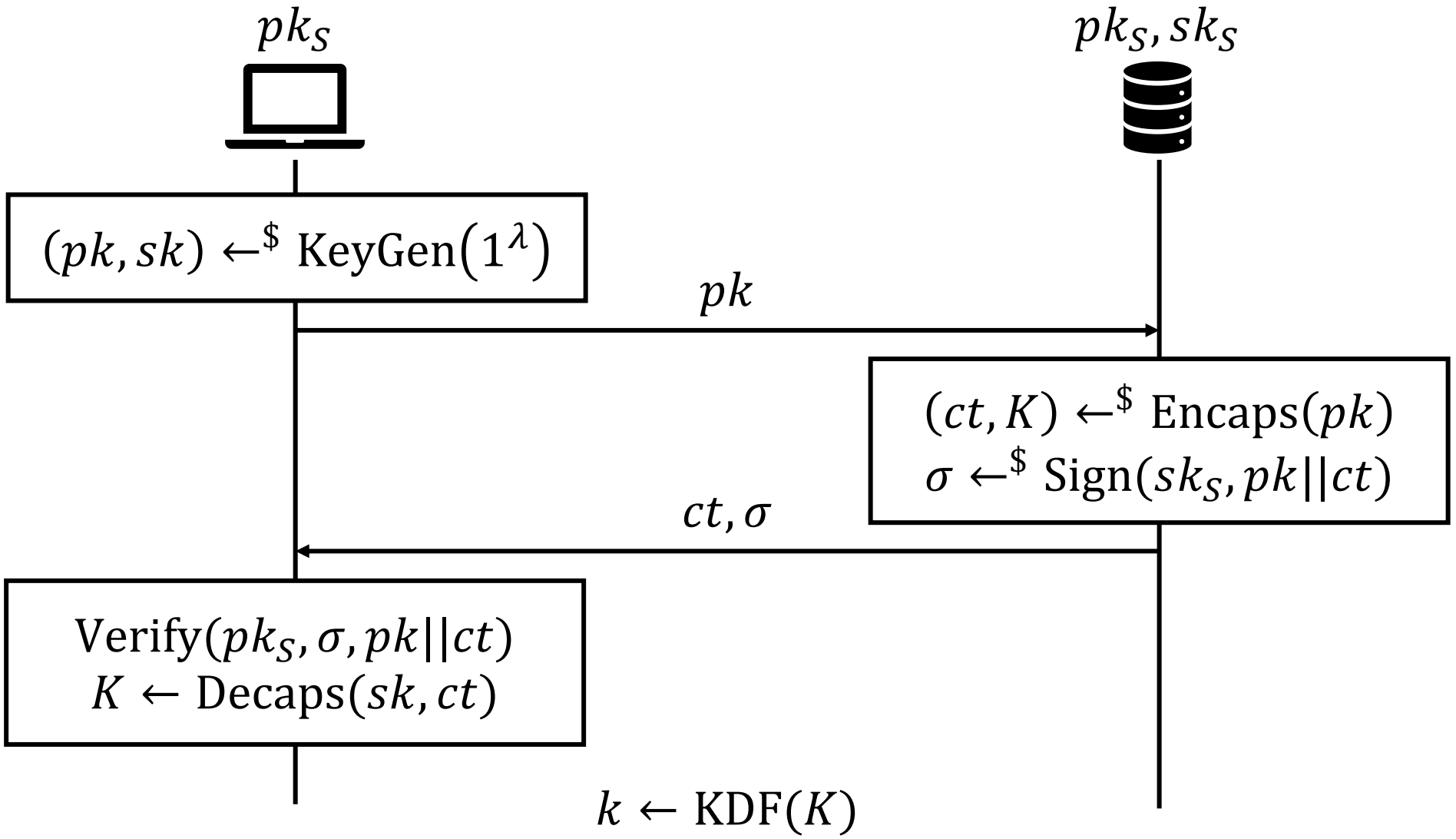


Better !

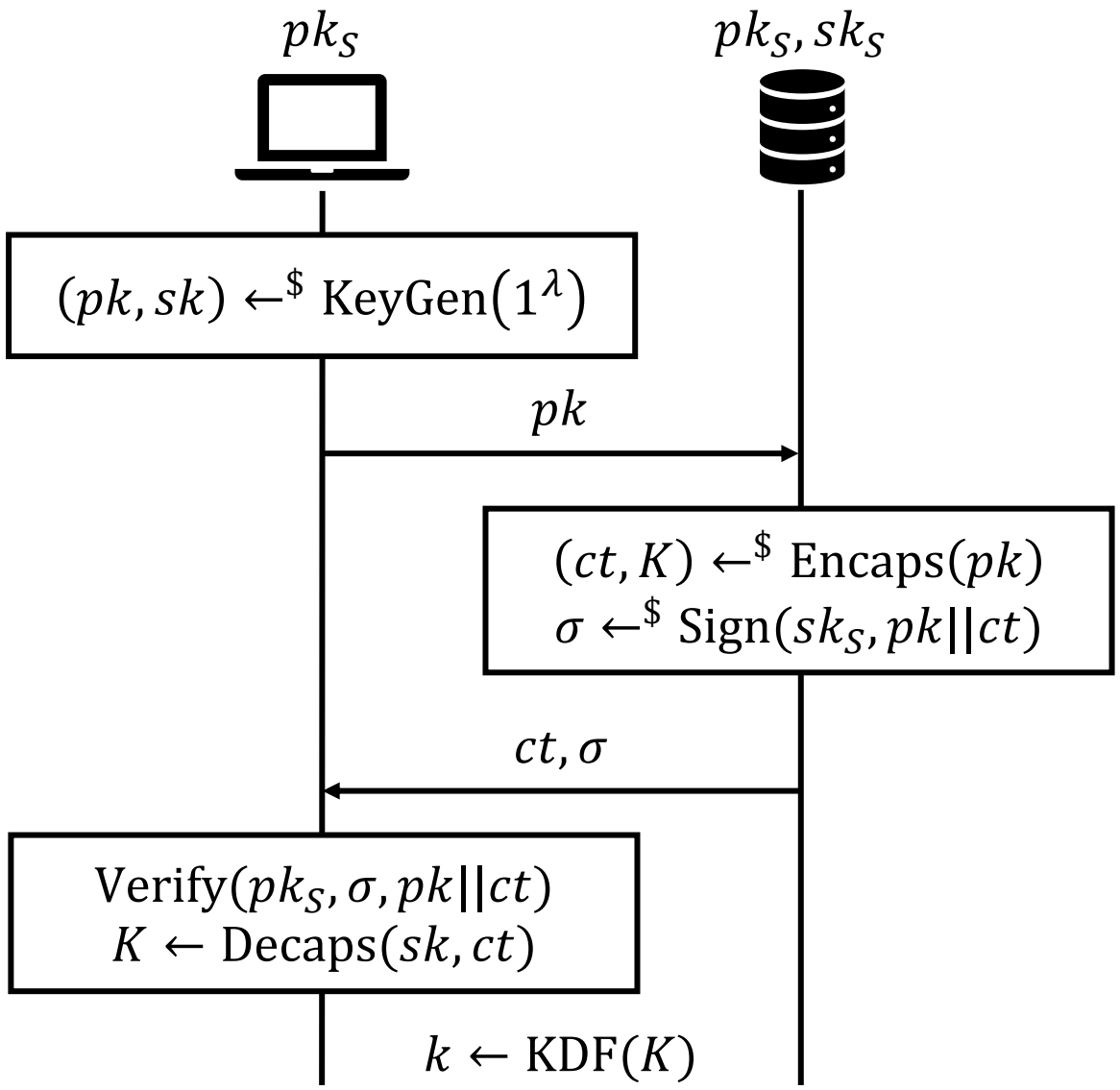
Compromise scenario C – Set of algorithms that can be subverted

Target protocol

Target protocol – The protocol

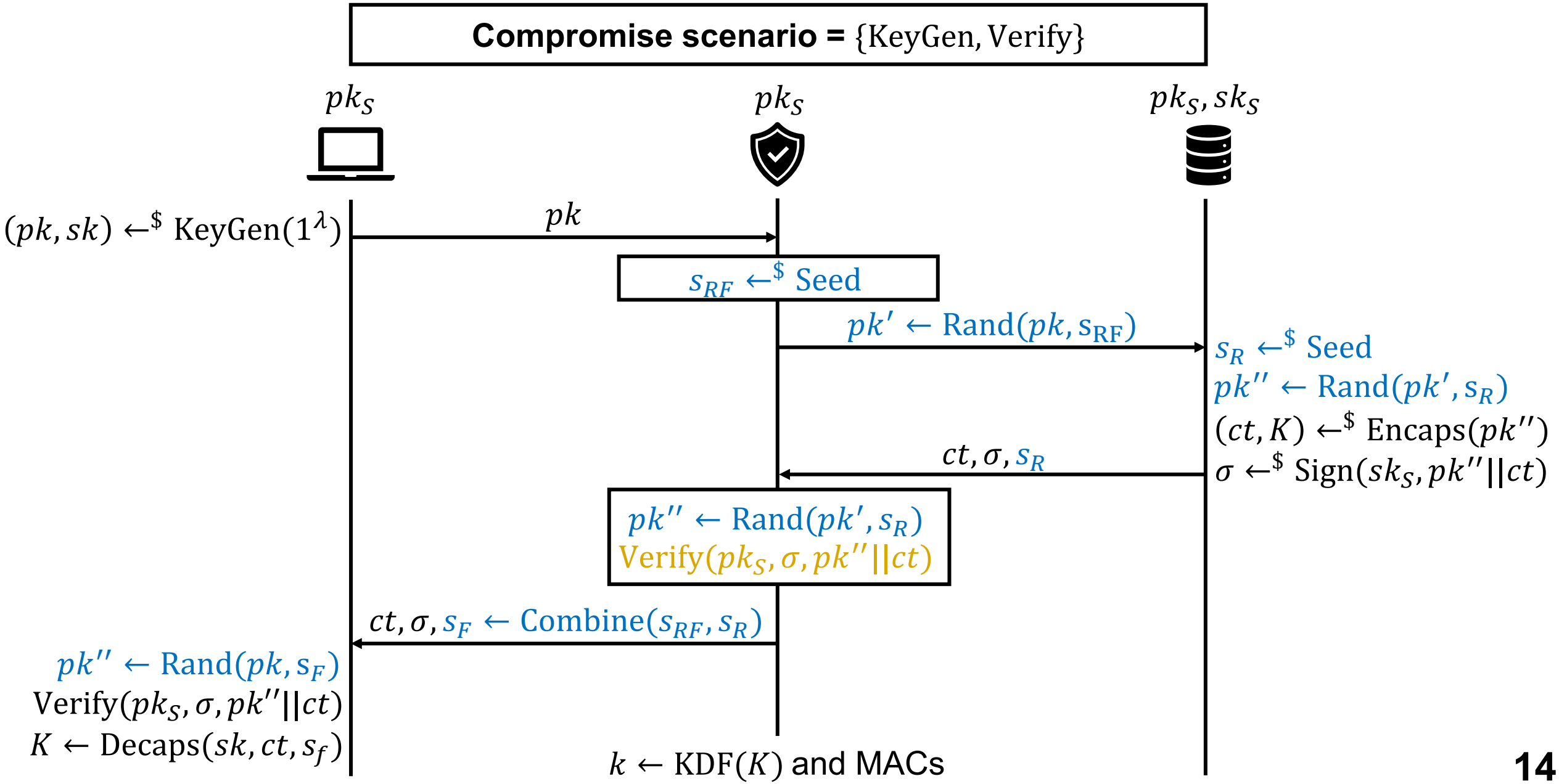


Target protocol – Protections



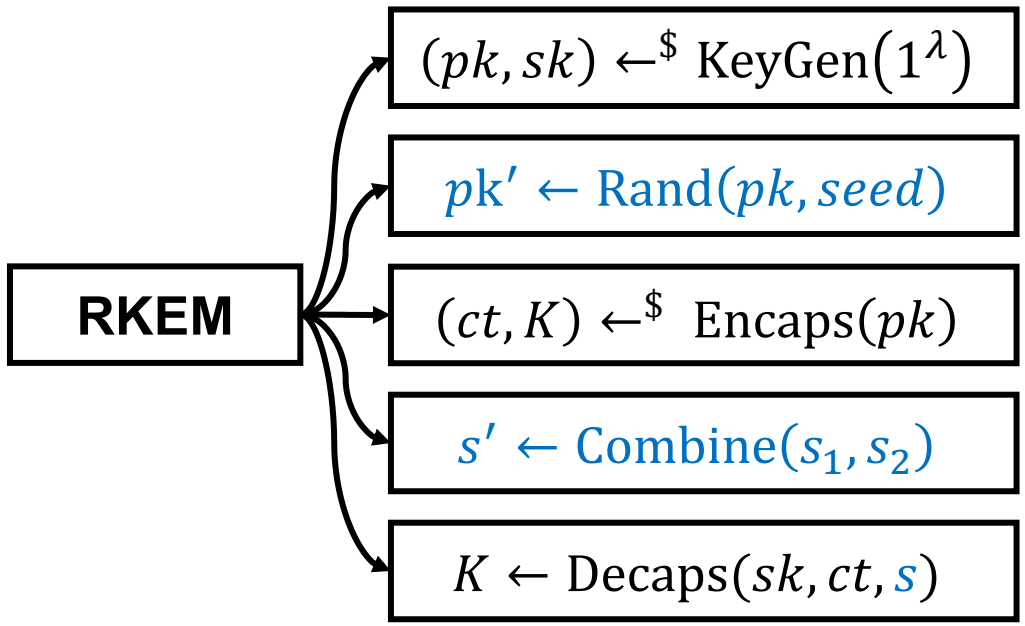
Target	RF action
Verify	Reverify the signature
KeyGen	Rerandomize the public key
Sign	Rerandomize the signature
Decaps, KDF	Double-layered encryption
Encaps	Very complicated

Target protocol – The protected protocol



A new primitive

New primitive – Definitions



RIND-CPA

$(pk, sk) \leftarrow^{\$} \mathrm{KeyGen}(1^{\lambda})$
$pk' \leftarrow \mathcal{A}(pk)$
$(ct, K_0) \leftarrow^{\$} \mathrm{Encaps}(pk')$
$K_1 \leftarrow^{\$} \mathcal{K}$
$b \leftarrow^{\$} \{0,1\}$
$d, seed \leftarrow \mathcal{A}(ct, K_b)$

$\mathcal{A}$ wins iff $b = d$ and $\mathrm{Rand}(pk, seed) = pk'$

RandIND

$pk_A \leftarrow \mathcal{A}()$
$s \leftarrow^{\$} \mathrm{Seed}$
return $\mathrm{Rand}(pk_A, s)$

≡

return $\mathrm{KeyGen}(1^{\lambda})$

New primitive – A DH instantiation

$$\text{KeyGen}(1^\lambda):\ x \leftarrow^{\$} \mathbb{F}_p,\ (\boldsymbol{pk}, \boldsymbol{sk}) \leftarrow (\boldsymbol{g^x}, \boldsymbol{x})$$

$$\text{Rand}(pk, seed):\ (\boldsymbol{pk'}) \leftarrow (\boldsymbol{pk^{seed}})$$

$$\text{Combine}(s_1, s_2):\ (\boldsymbol{s'}) \leftarrow (\boldsymbol{s_1} \times \boldsymbol{s_2})$$

$$\text{Encaps}(pk):\ y \leftarrow^{\$} \mathbb{F}_p,\ (\boldsymbol{ct}, \boldsymbol{K}) \leftarrow \left(\boldsymbol{g^y}, \boldsymbol{H}(\boldsymbol{pk^y})\right)$$

$$\text{Decaps}(sk, ct, seed):\ (\boldsymbol{K}) \leftarrow \left(\boldsymbol{H}\left(\boldsymbol{ct^{sk \times seed}}\right)\right)$$

Check that $seed \neq 0$ and $seed \neq 1$

New primitive – A post-quantum instantiation

$$\text{KeyGen}(1^\lambda):$$
$$(s, e) \leftarrow^{\$} \chi_s^{2k}$$
$$(\boldsymbol{pk}, \boldsymbol{sk}) \leftarrow (\boldsymbol{As} + \boldsymbol{e}, (\boldsymbol{s}, \boldsymbol{e}))$$

$$\text{Rand}(pk, seed = (s^*, e^*)):$$
$$pk^* \leftarrow As^* + e^*$$
$$(\boldsymbol{pk'}) \leftarrow (\boldsymbol{pk} + \boldsymbol{pk}^*)$$

$$\text{Combine}((s_1, e_1), (s_2, e_2)):$$
$$(\boldsymbol{s'}, \boldsymbol{e'}) \leftarrow (\boldsymbol{s_1} + \boldsymbol{s_2}, \boldsymbol{e_1} + \boldsymbol{e_2})$$

$$\text{Encaps}(pk):$$
$$\mu \leftarrow^{\$} \{0,1\}^*$$
$$(u, v) \leftarrow \text{Kyber.Enc}(pk, \mu)$$
$$(\boldsymbol{ct}, \boldsymbol{K}) \leftarrow ((\boldsymbol{u}, \boldsymbol{v}), \boldsymbol{H}(\boldsymbol{\mu}))$$

$$\text{Decaps}((s, e), (u, v), (s^*, e^*)):$$
$$\mu \leftarrow \text{Kyber.Dec}(s + s^*, (u, v))$$
$$(\boldsymbol{K}) \leftarrow (\boldsymbol{H}(\boldsymbol{\mu}))$$

* Check that *seed* is not too big

Proofs (pen-and-paper + cryptoverif)

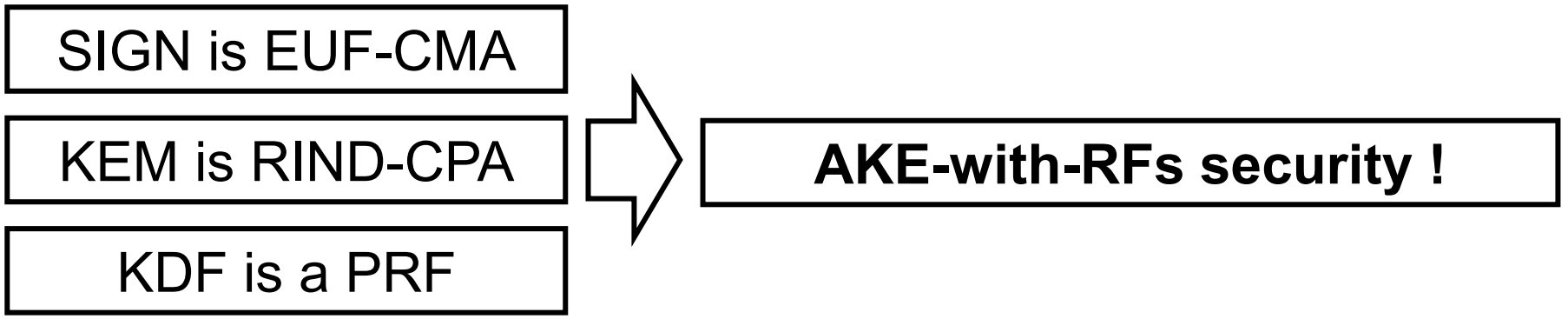
Protocol proofs – Bad RF case

Usual **AKE-security**

Oracles : NewInstance, Send, Reveal, Corrupt, Test

AKE-with-RFs

+ NewInstance1, NewRF, SetupRF
[warning] : Setup-free RFs



Protocol proofs – Security preservation

Authenticating : restore authentication (lost if σ not verified / pk collision)

Securing : restore key-secrecy (lost if weak pk)

Oracles : NewInstance1, Send, Reveal, Corrupt, Test, SetupRF, SetCode

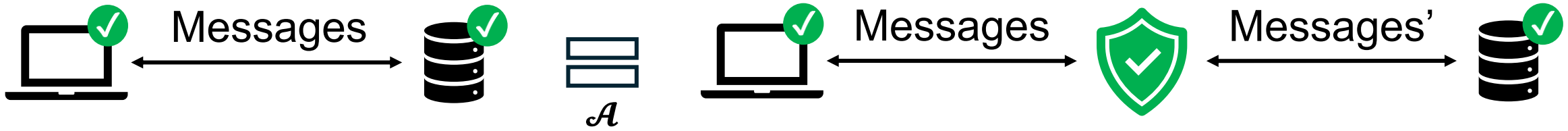
SIGN is EUF-CMA

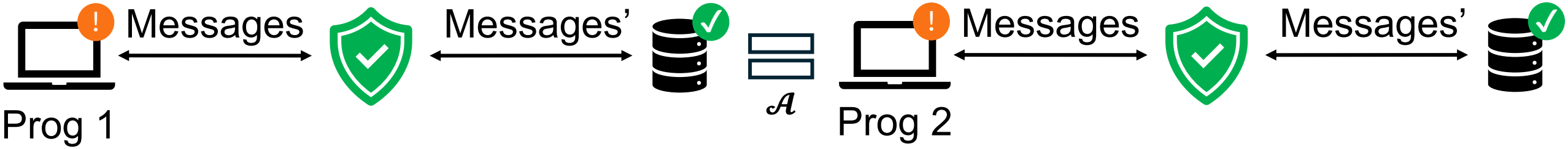
KEM is RIND-CPA & RandIND

KDF is a PRF

⇒ **Authenticating & Securing !**

Protocol security – Other notions





SIGN is EUF-CMA

KEM is RandIND

Obliviousness & Exf-res !

Conclusion

Conclusion

1. Setup-free RFs, relaxations, compromise scenario ✓
2. PQ-TLS RF for Verify and Keygen ✓
3. Rerandomizable KEM ✓
4. Cryptoverif proofs ✓

Thank you for your attention !

Questions ?

CryptoVerif proofs

AKE-with-RFs (setup-free), authenticating, securing, obliviousness

Usual IND-CPA

$\boldsymbol{Opk}()$: Return honest pk[i] $\boldsymbol{OE}(\boldsymbol{pk})$: Return Encaps(pk)	$\boldsymbol{Opk}()$: Return honest pk[i] $\boldsymbol{OE}(\boldsymbol{pk})$: pk honest : return (ct, rand K) otherwise : return Encaps(pk)

RIND-CPA

Oracles :

Oracle 1 : Opk[i]() -> pk

Oracle 2 : OE[ie](pk) -> ct

Oracle 3 : OS[ie,is](pk, seed) -> K

Ideas :

Can't send K in OE : seed is unknown !

OS : check rand(pk, seed) = OE[ie].pk

Authenticating + securing

Ideas :

Fuse client + RF

Signature = RF signature

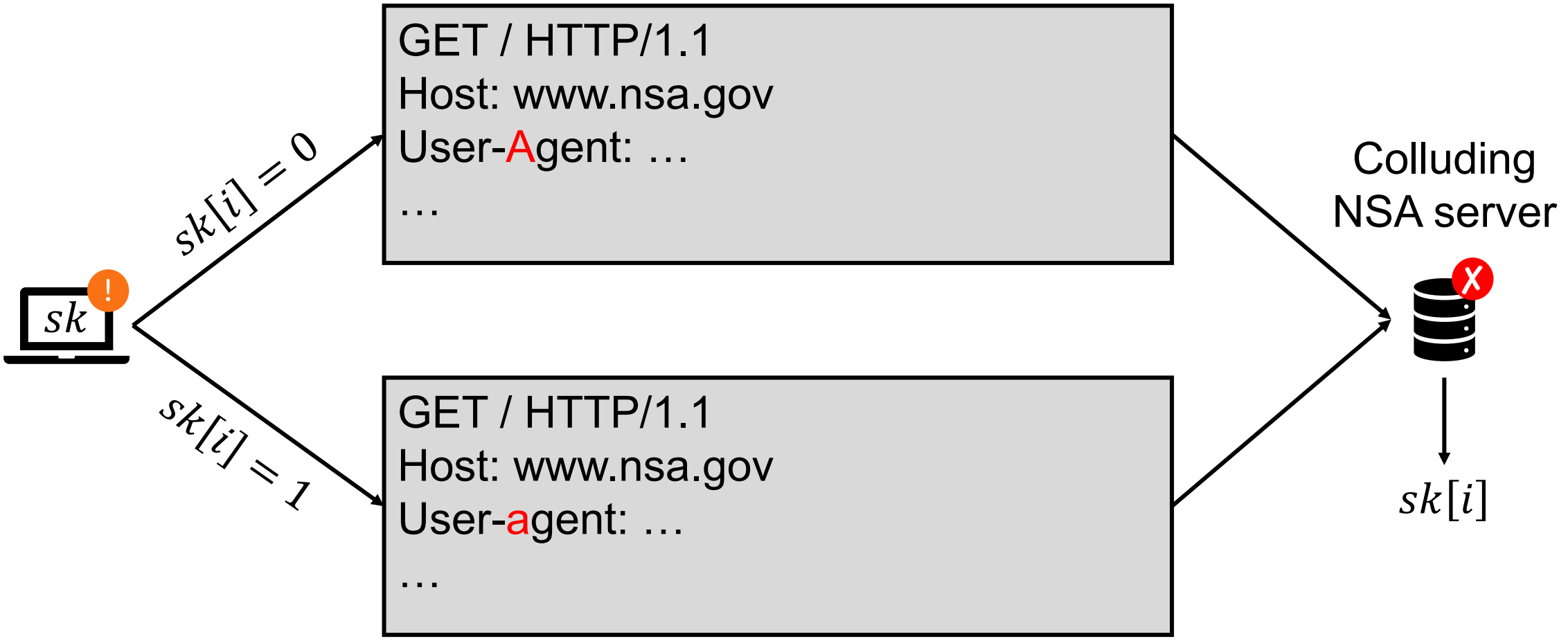
PK = « given by adversary »

Obliviousness

Ideas :

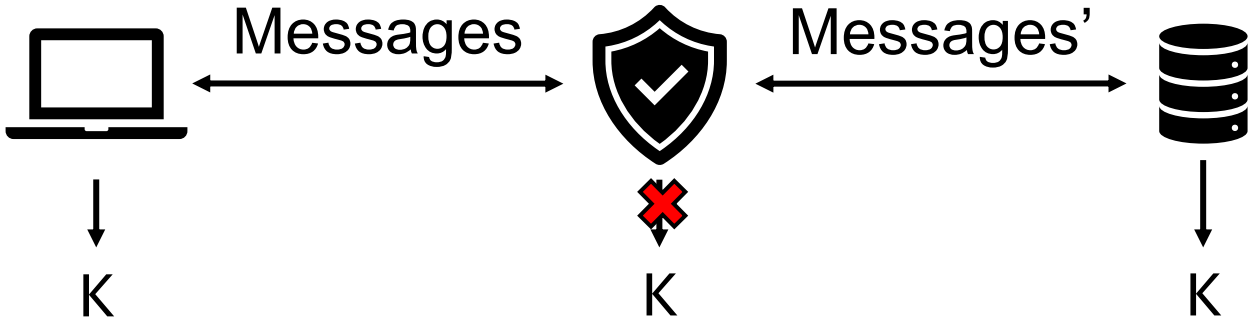
Equivalence between C and C+RF fused

No malicious server – exf-res and obliviousness

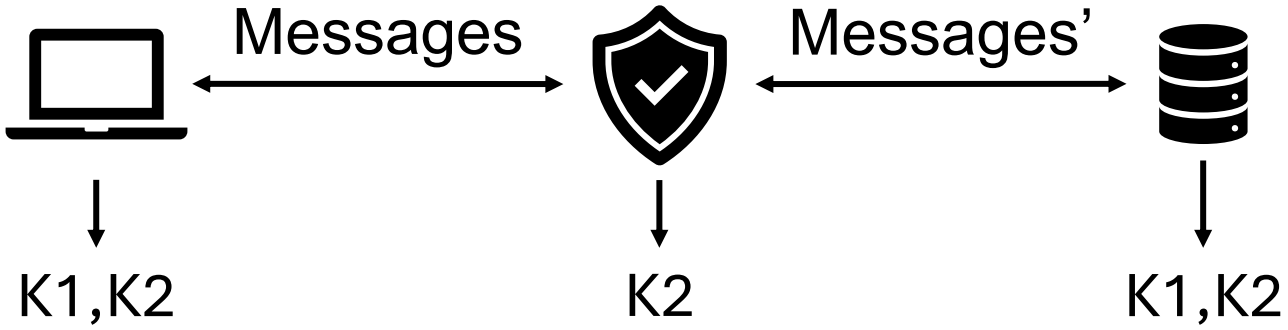


Double-layer encryption trick

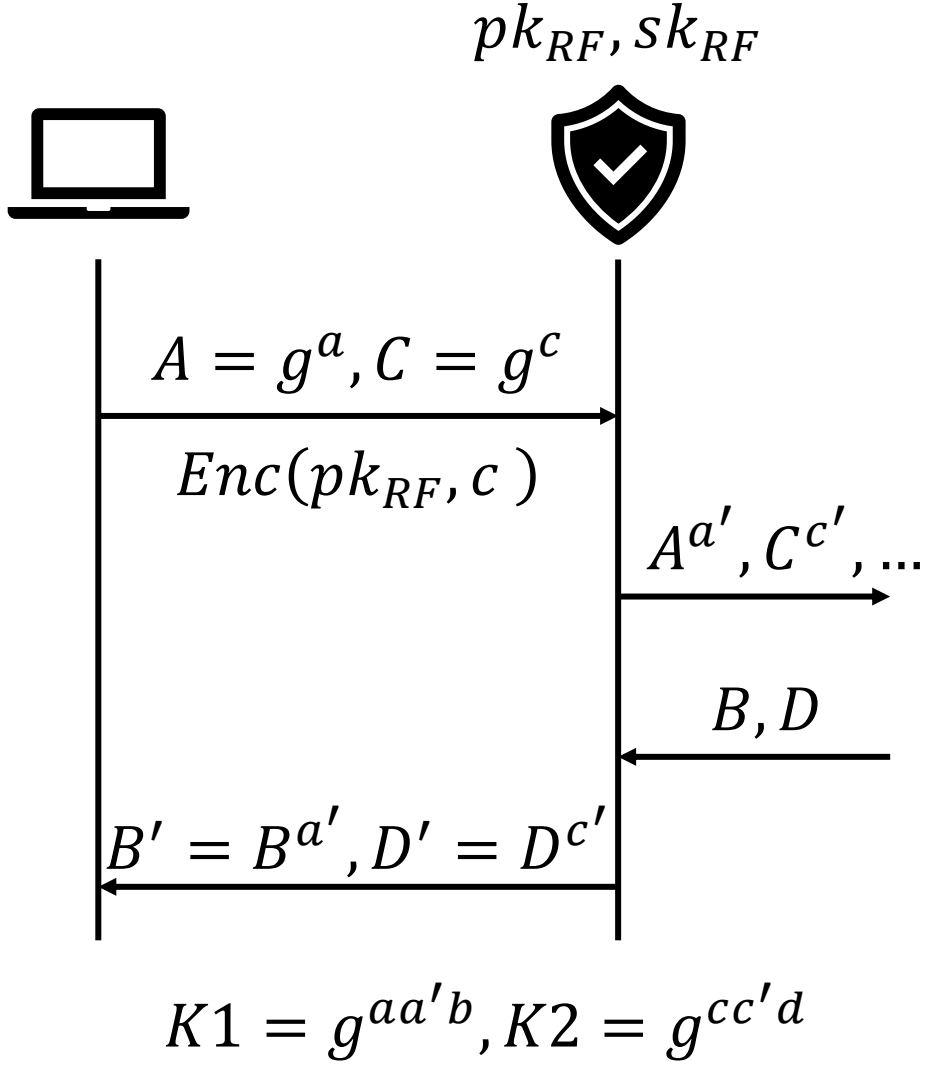
One key case



Two key case



How it works



$$K1 = g^{aa'b}, K2 = g^{cc'd}$$