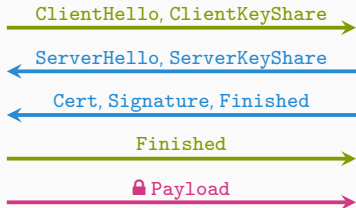


Bloom Filter Encryption and Applications to Efficient Forward-Secret o-RTT Key Exchange

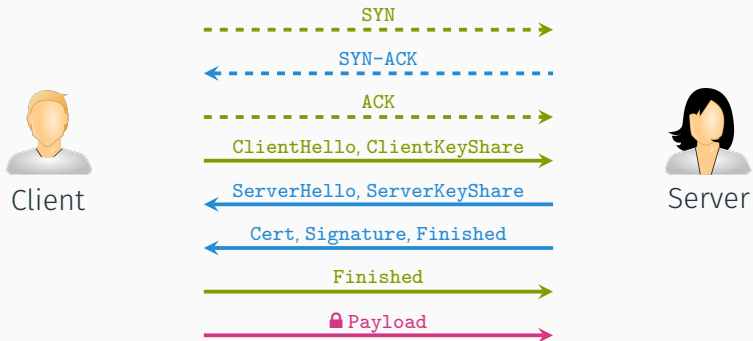
David Derler[‡], Tibor Jager^{||}, Daniel Slamanig[§], Christoph Striecks[§]
January 12, 2018—RWC 2018, Zurich, Switzerland



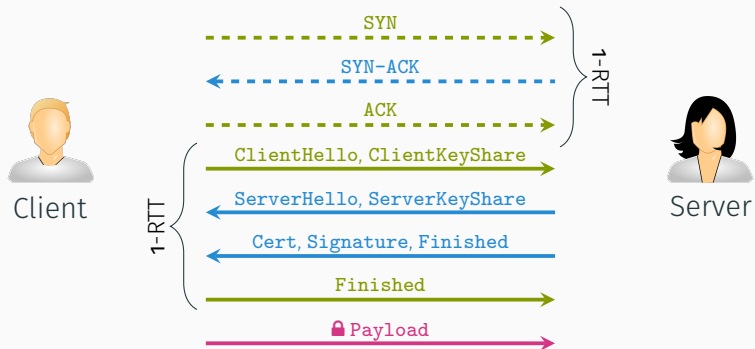
Key Establishment with TLS



Key Establishment with TLS

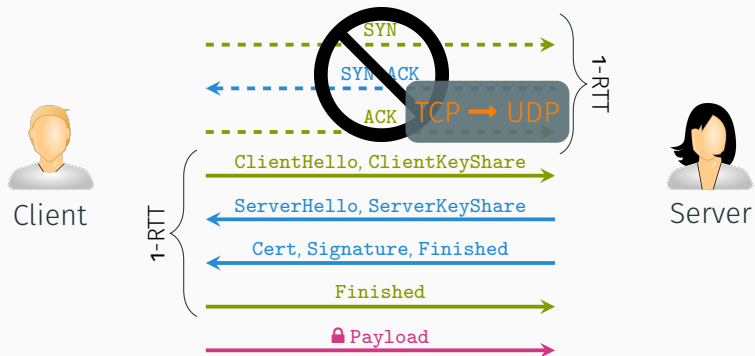


Key Establishment with TLS



- > 2-RTTs before first payload message
- ? Is this necessary

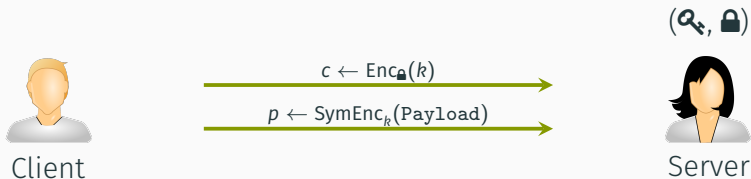
Key Establishment with TLS



- > 2-RTTs before first payload message
- ? Is this necessary

We want to send *cryptographically protected*
payload in *first message* (0-RTT KE)

Trivial Protocol



Major deficiencies:

- No forward secrecy
- Vulnerable to replay attacks

0-RTT in TLS1.3/QUIC

- First session 1-RTT, session resumption 0-RTT
- ✓ Replay protection
- ? Forward secrecy for most transmitted data

o-RTT in TLS1.3/QUIC

- First session 1-RTT, session resumption o-RTT
- ✓ Replay protection
- ? Forward secrecy for most transmitted data

Full forward secrecy, replay protection, and o-RTT?

- A priori not even clear if possible
- 📖 Günther, Hale, Jager, and Lauer at EUROCRYPT'17
- Using puncturable encryption (Green, Miers. S&P 2015)

Puncturable Encryption

Conventional encryption scheme:

- (KeyGen, Enc, Dec)
- + Additional algorithm $Q'_k \leftarrow \text{Punc}(Q_k, C)$

Properties

- Q'_k no longer useful to decrypt C
- Q'_k still useful to decrypt other ciphertexts
- Repeated puncturing possible

Puncturable Encryption


Conventional encryption scheme:

- (KeyGen, Enc, Dec)
- + Additional algorithm $Q'_k \leftarrow \text{Punc}(Q_k, C)$

Properties

- Q'_k no longer useful to decrypt C
- Q'_k still useful to decrypt other ciphertexts
- Repeated puncturing possible

o-RTT KE via Puncturable Encryption

- Client encrypts message under public key 
- Server decrypts using secret key Q'_k
- Server punctures Q'_k on C

Downsides of existing approaches

- Puncturing and/or decryption expensive
(experiments by authors of [GHJL17]: 30s - several minutes)

Downsides of existing approaches

- Puncturing and/or decryption expensive
(experiments by authors of [GHJL17]: 30s - several minutes)

Observation

- Can accept somewhat larger (secret) keys
- Can accept non-negligible correctness error
- For example, 1 in 1000 sessions fail
- › Can fall back to 1-RTT in this case

Bloom Filters



- Initial state $T := \mathbf{0}^m$
- k universal hash functions $(H_j)_{j \in [k]}$
- $H_j : \mathcal{U} \rightarrow [m]$
- Throughout this talk, let $k = 3$

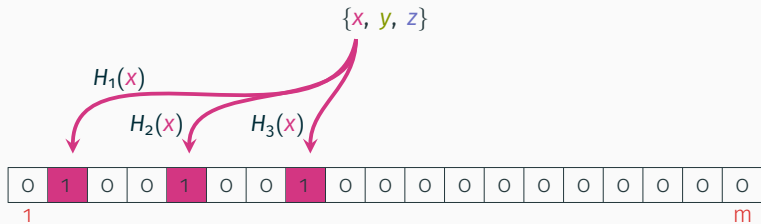
Bloom Filters

$\{x, y, z\}$

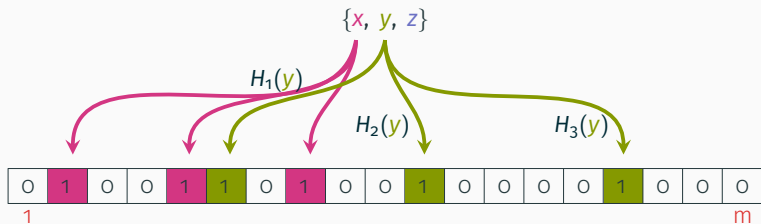


- Initial state $T := \mathbf{0}^m$
- k universal hash functions $(H_j)_{j \in [k]}$
- $H_j : \mathcal{U} \rightarrow [m]$
- Throughout this talk, let $k = 3$

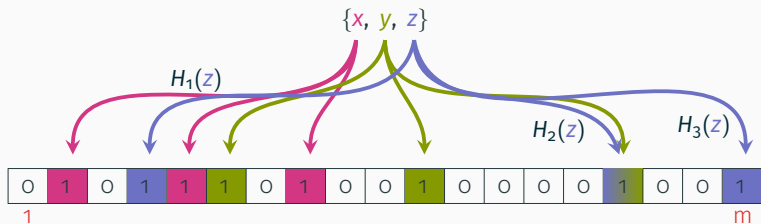
Bloom Filters



Bloom Filters



Bloom Filters



Properties

- No false negatives

Bloom Filters

{x, y, z}



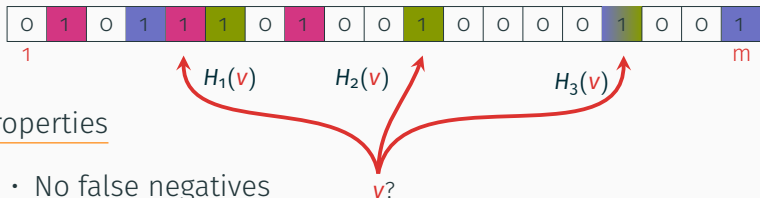
Properties

- No false negatives

w?

Bloom Filters

{x, y, z}

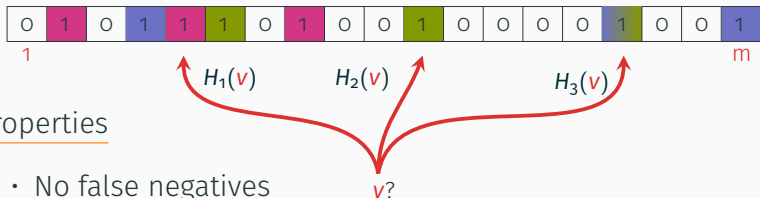


Properties

- No false negatives
- False positives possible

Bloom Filters

{x, y, z}



Properties

- No false negatives
- False positives possible
- Probability determined by k , m , and # inserted elements

Bloom Filter Encryption



Setup

- Set up BF

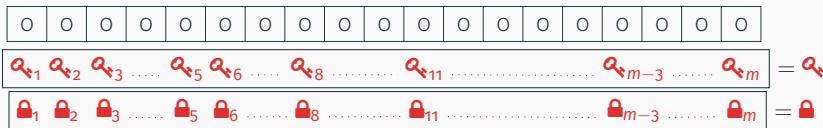
Bloom Filter Encryption



Setup

- Set up BF
- Associate key pair to each bit

Bloom Filter Encryption



Setup

- Set up BF
- Associate key pair to each bit
- Compose BFE key pair (Q, L)

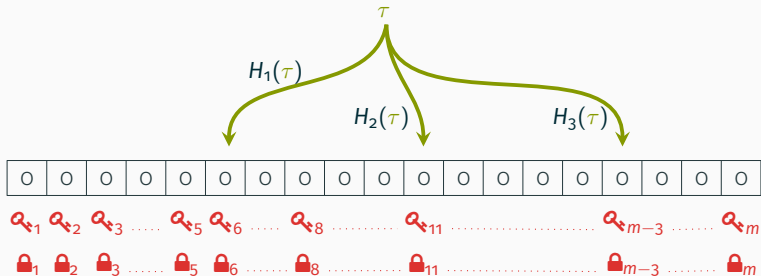
Bloom Filter Encryption



Encrypt message M

- Randomly choose tag τ

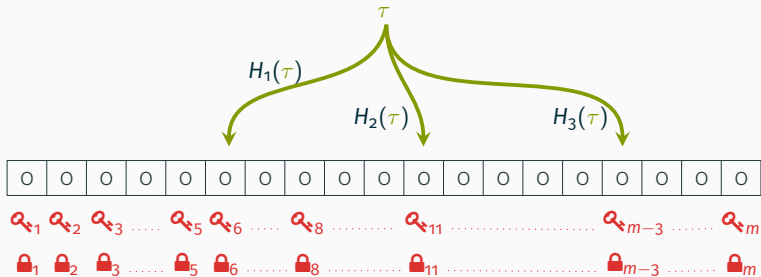
Bloom Filter Encryption



Encrypt message M

- Randomly choose tag τ
- Determine indexes from τ

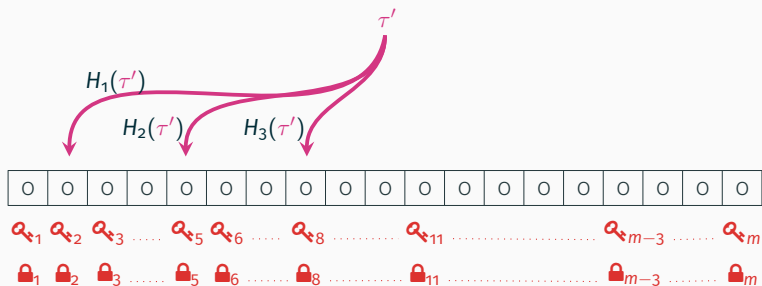
Bloom Filter Encryption



Encrypt message M

- Randomly choose tag τ
- Determine indexes from τ
- $C_\tau \leftarrow \text{Enc}_{\text{padlock}_6 \vee \text{padlock}_{11} \vee \text{padlock}_{m-3}}(M)$

Bloom Filter Encryption

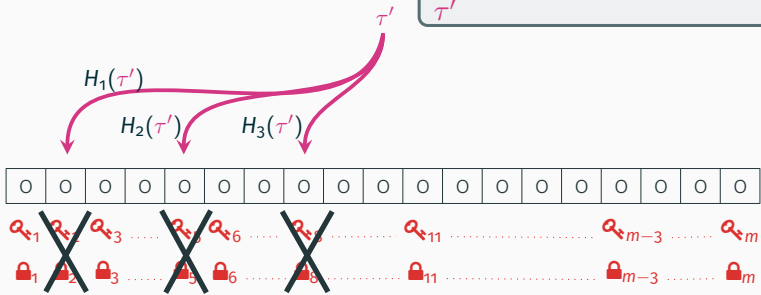


Puncture ciphertext $C_{\tau'}$

- Determine BF indexes from τ'

Bloom Filter Encryption

i Secret key no longer useful to decrypt $C_{\tau'}$ with associated tag τ'

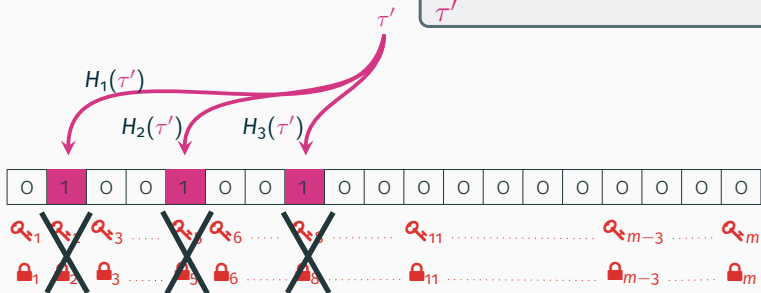


Puncture ciphertext $C_{\tau'}$

- Determine BF indexes from τ'
- Delete associated keys

Bloom Filter Encryption

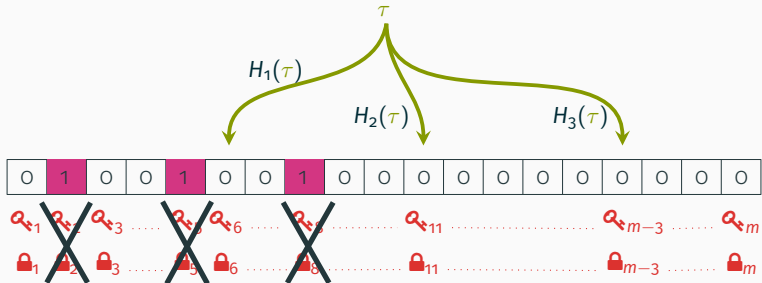
i Secret key no longer useful to decrypt $C_{\tau'}$ with associated tag τ'



Puncture ciphertext $C_{\tau'}$

- Determine BF indexes from τ'
- Delete associated keys
- Update BF state

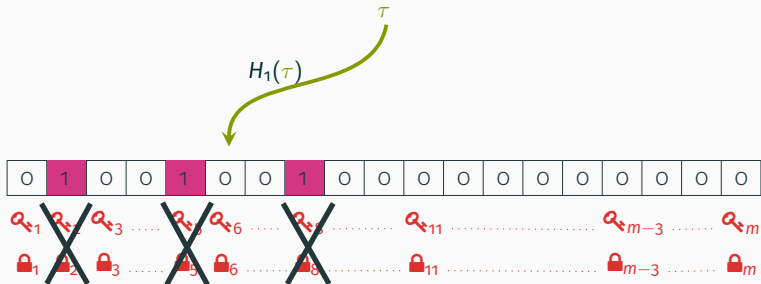
Bloom Filter Encryption



Decrypt ciphertext C_τ

- Determine BF indexes from τ

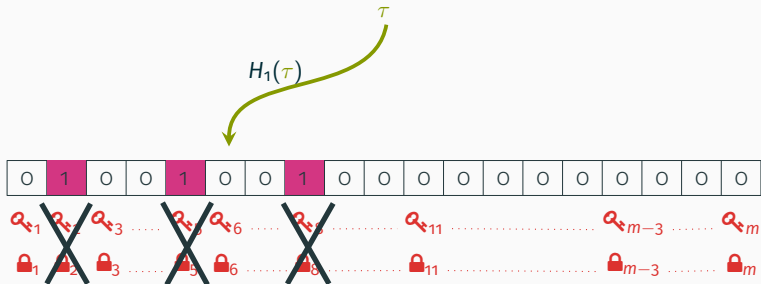
Bloom Filter Encryption



Decrypt ciphertext C_τ

- Determine BF indexes from τ
- Let i lowest index w. $BF[i] = 0$

Bloom Filter Encryption



Decrypt ciphertext C_τ

- Determine BF indexes from τ
- Let i lowest index w. $BF[i] = 0$
- $M \leftarrow \text{Dec}_{q_{\tau_i}}(C_\tau)$

Example BF Parameters


We let

- Maximum # of elements in BF: 2^{20}
- $\approx 2^{12}$ puncturings/day for full year
- False positive probability: 10^{-3}


Then we get

- BF size $m = n \ln p / (\ln 2)^2 \approx 2MB$
- # hash functions $k = \lceil m / n \ln 2 \rceil = 10$


Based on Boneh-Franklin IBE

- Constant size public key (400 bit at 120 bit security)
- Identity per BF position
- Secret key: include one IBE- per identity

Based on Boneh-Franklin IBE

- Constant size public key (400 bit at 120 bit security)
 - Identity per BF position
 - Secret key: include one IBE- per identity
 - Ciphertext
 - > k Boneh-Franklin ciphertexts w. shared rand.
 - > Use hashed variant to save space
 - > Size $\mathcal{O}(k)$
- \approx 3000 bit (120 bit security, parameters from before)

Based on Boneh-Franklin IBE

- Constant size public key (400 bit at 120 bit security)
- Identity per BF position
- Secret key: include one IBE- per identity
- Ciphertext
 - > k Boneh-Franklin ciphertexts w. shared rand.
 - > Use hashed variant to save space
 - > Size $\mathcal{O}(k)$ \approx 3000 bit (120 bit security, parameters from before)
- Secret key size \approx 700MB (parameters from before)

CCA security?

- Requires additional technicalities
- Details in the paper (EUROCRYPT'18; preprint will follow soon)

CCA security?

- **Requires additional technicalities**
- Details in the paper (EUROCRYPT'18; preprint will follow soon)

Constant size ciphertexts?

- Adaptively secure small universe ABE
- Constant size ciphertexts ABE

Extensions

- Time-based BFE (TBBFE)
- Enable multiple time intervals
- Similar approach as [GM15,GHJL17]

Extensions

- Time-based BFE (TBBFE)
- Enable multiple time intervals
- Similar approach as [GM15,GHJL17]

Other instantiations?

- Potentially many other possible instantiations
(work in progress)

Existing approaches

- Most critical ops expensive (puncturing & decryption)
 - ! Authors of [GHJL17] report 30s to minutes

Conclusions

Existing approaches

- Most critical ops expensive (puncturing & decryption)
 - ! Authors of [GHJL17] report 30s to minutes

Our approach

- ✓ Offload expensive ops to less critical phases
(key generation, resp. switch of time interval for TB)
- ✓ Very efficient decryption (\approx ElGamal in \mathbb{G}_T)
- ✓ Only deletions & hash evaluations upon puncture
- ✓ Conjectured dec. & punc. times in order of milliseconds
- ✓ Applications of BFE beyond o-RTT KE

Next steps?

- > Real world implementation and deployment



@dderler

@tibor_jager

@drl3c7er

@CStriecks