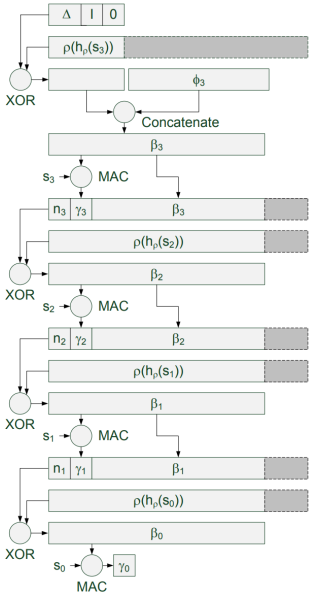
Sphinx Packets

Decentralized Header Construction

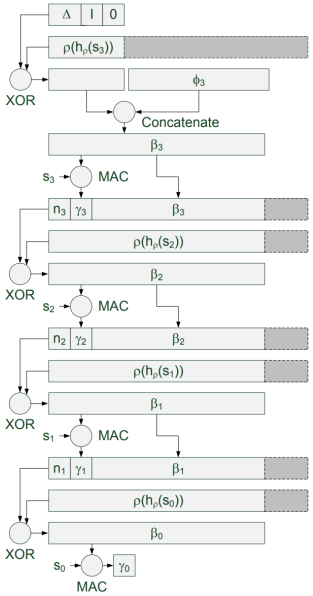
Aurélien Chassagne

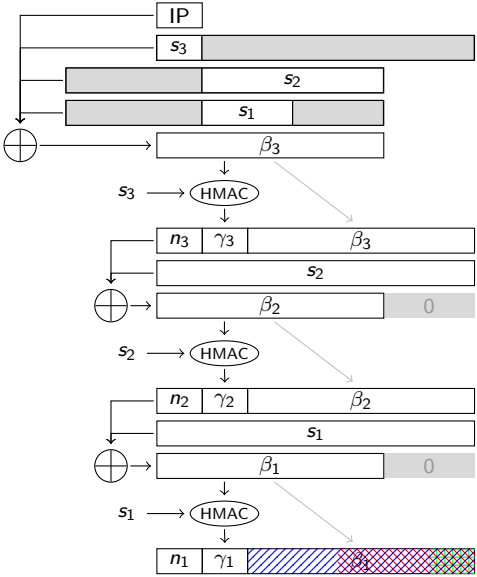
February 18, 2025

Reproduce article's figure

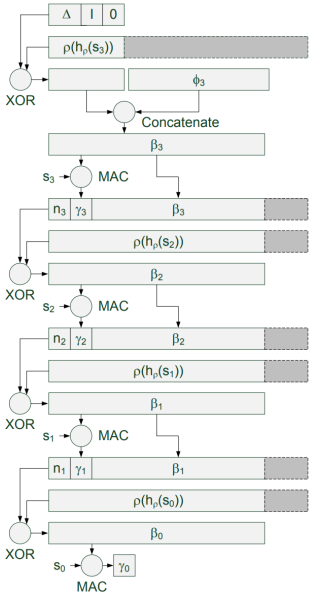


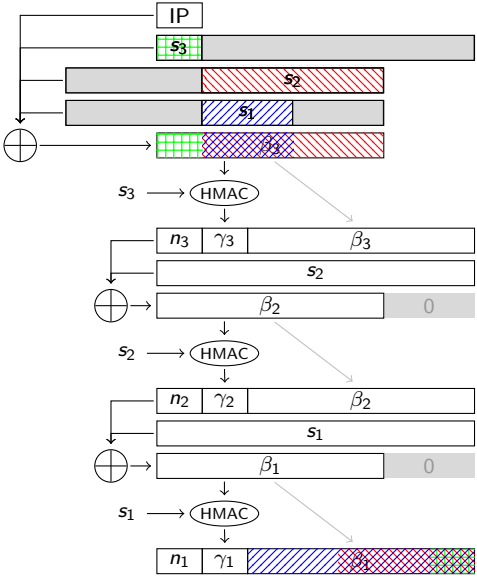
Reproduce article's figure



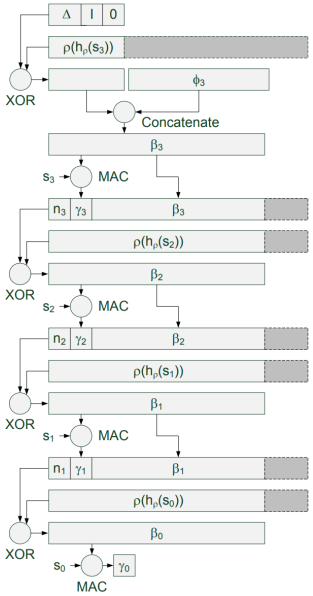


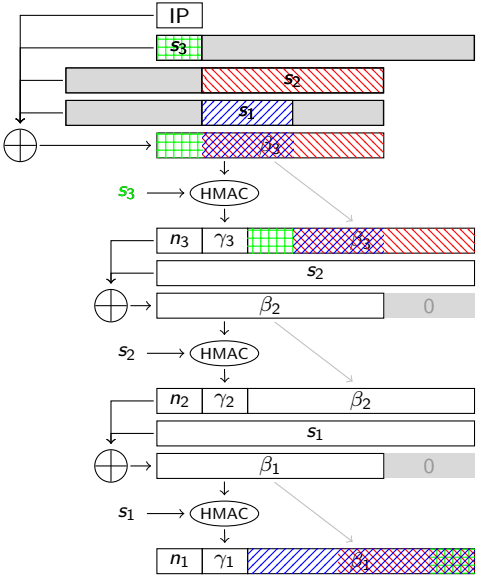
Reproduce article's figure



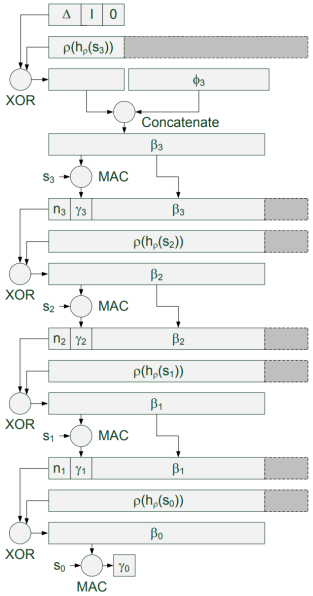


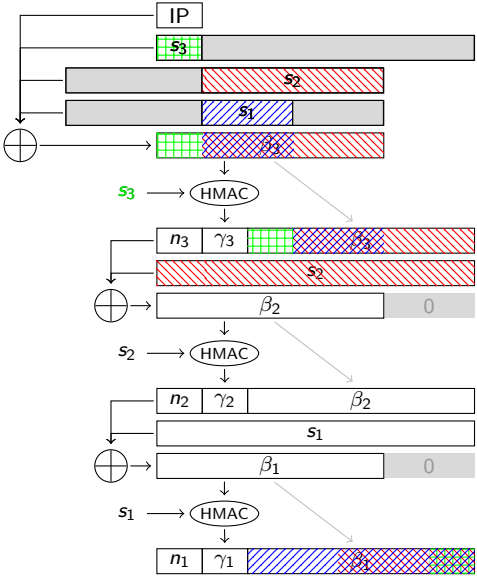
Reproduce article's figure



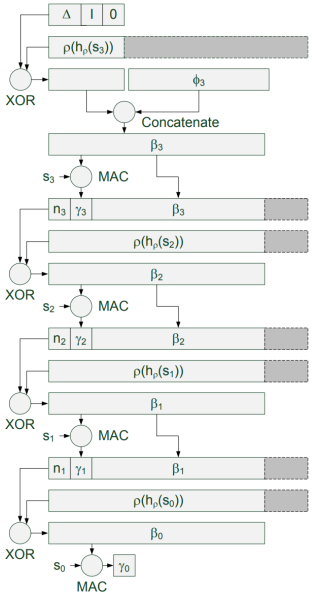


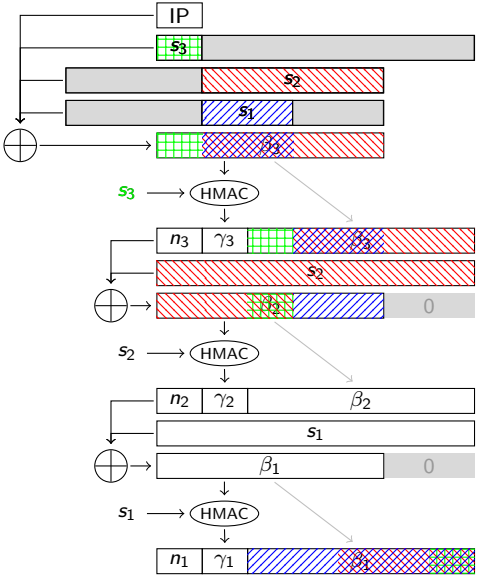
Reproduce article's figure



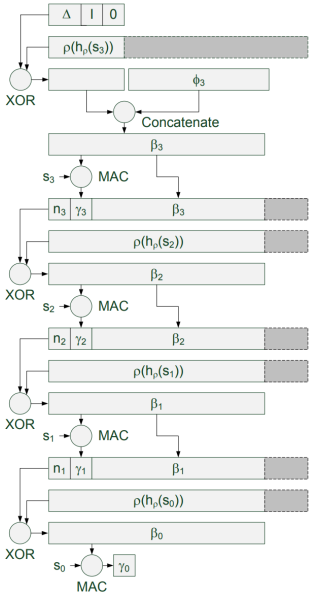


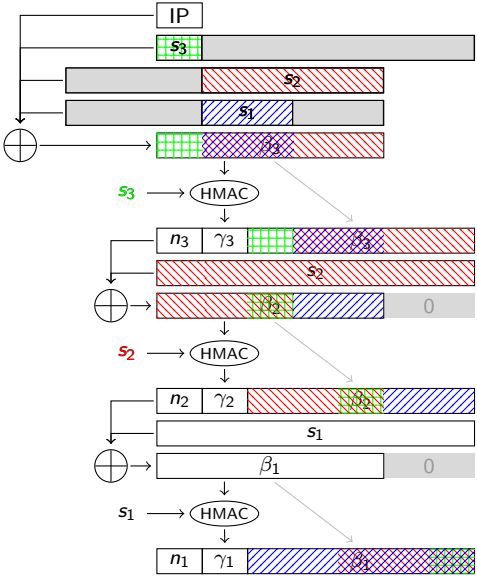
Reproduce article's figure



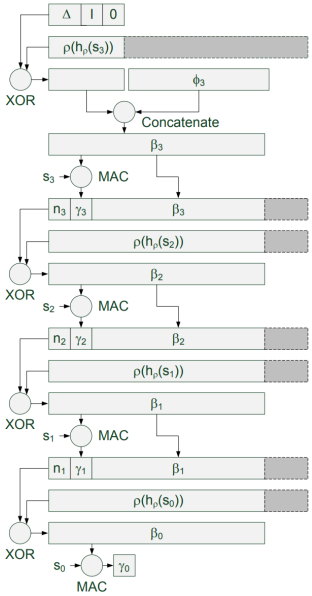


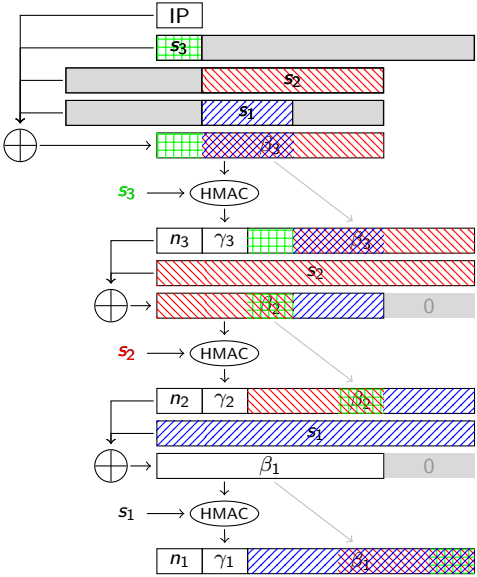
Reproduce article's figure



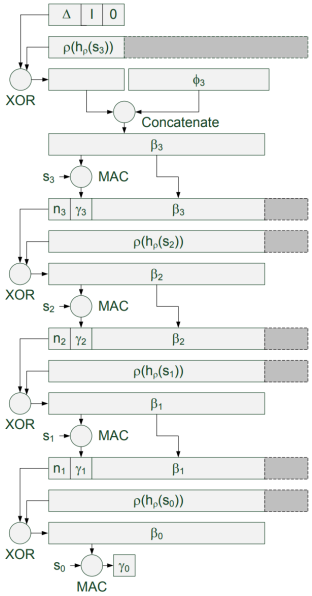


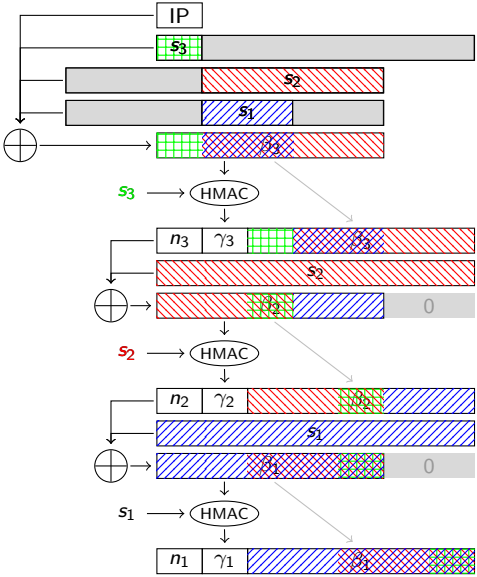
Reproduce article's figure



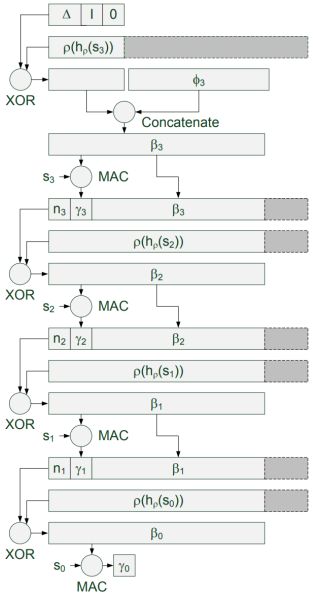


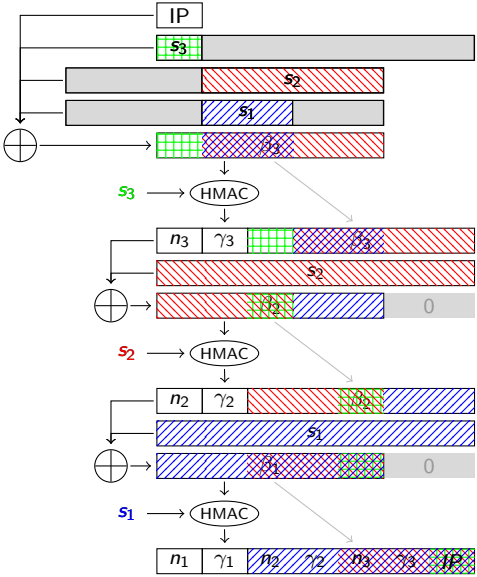
Reproduce article's figure



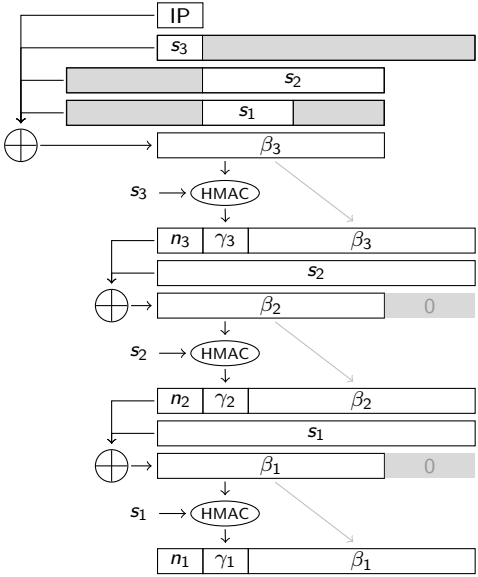


Reproduce article's figure



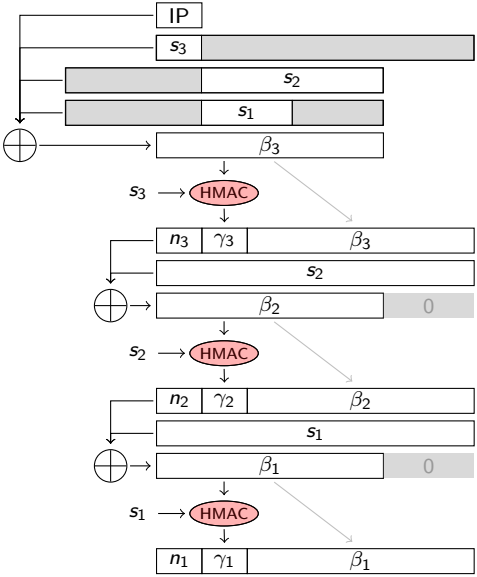


Adapting HMAC



Adapting HMAC

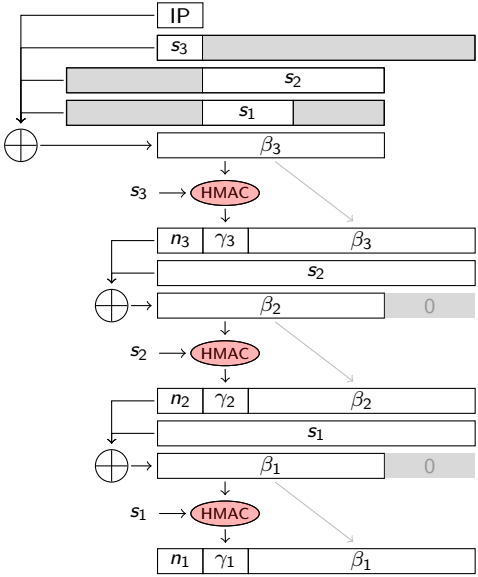
Main problem: Decentralizing a **Hash** ?



Adapting HMAC

Main problem: Decentralizing a **Hash** ?

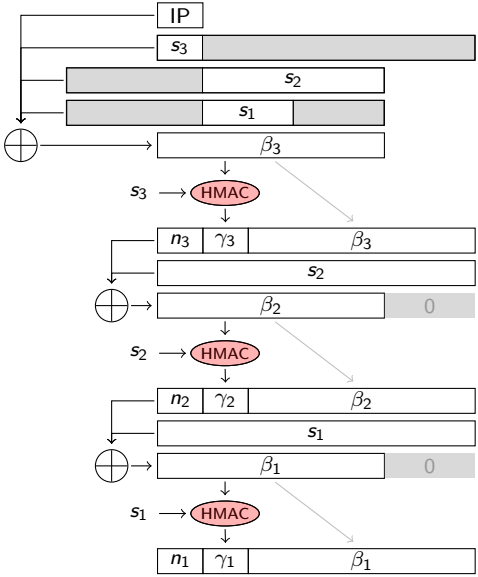
- Need **homomorphic properties** to split computation and aggregate results.



Adapting HMAC

Main problem: Decentralizing a **Hash** ?

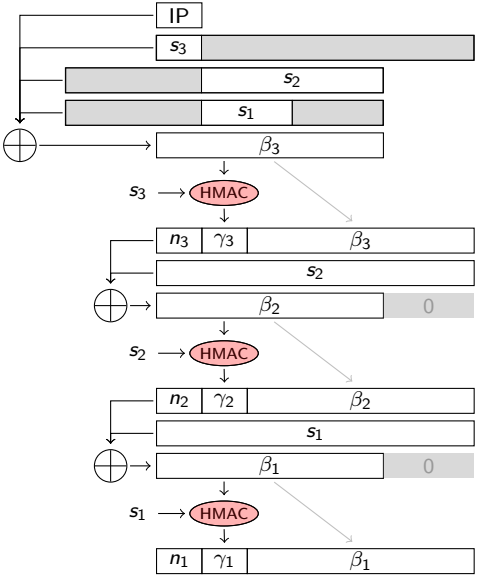
- Need **homomorphic properties** to split computation and aggregate results.
- Secure homomorphic hash **seems impractical** (no promising solutions found).



Adapting HMAC

Main problem: Decentralizing a **Hash** ?

- Need **homomorphic properties** to split computation and aggregate results.
- Secure homomorphic hash **seems impractical** (no promising solutions found).
- Exploring **homomorphic encryption** as an alternative.

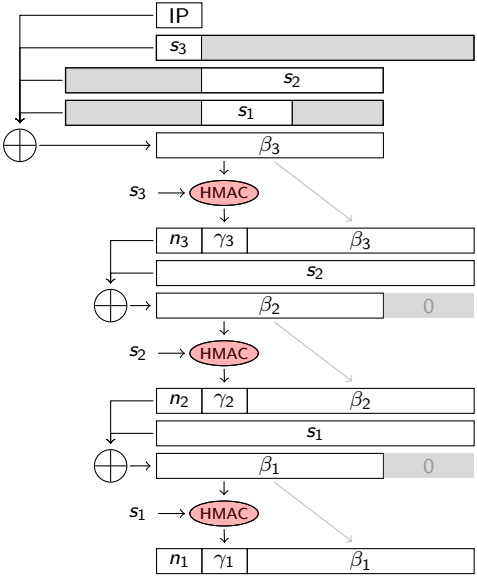


Adapting HMAC

Main problem: Decentralizing a **Hash** ?

- Need **homomorphic properties** to split computation and aggregate results.
- Secure homomorphic hash **seems impractical** (no promising solutions found).
- Exploring **homomorphic encryption** as an alternative.

1. **RSA**
2. **ElGamal**
3. **Paillier**



Adapting HMAC

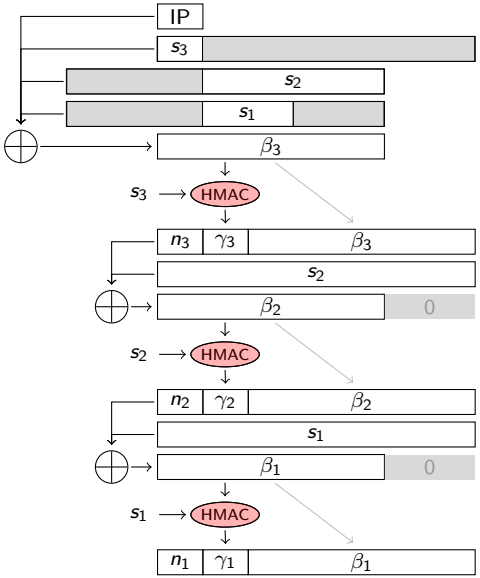
Main problem: Decentralizing a **Hash** ?

- Need **homomorphic properties** to split computation and aggregate results.
- Secure homomorphic hash **seems impractical** (no promising solutions found).
- Exploring **homomorphic encryption** as an alternative.

1. **RSA**
2. **ElGamal**
3. **Paillier**

$$\begin{aligned}\mathcal{E}(m_1)\cdot\mathcal{E}(m_2) &= (g^{m_1} r_1^n)(g^{m_2} r_2^n) \bmod n^2 \\ &= g^{m_1+m_2}(r_1 r_2)^n \bmod n^2 \\ &= \mathcal{E}(m_1+m_2).\end{aligned}$$

Problem: Mix of different operations... order matters!



Adapting HMAC

Main problem: Decentralizing a **Hash** ?

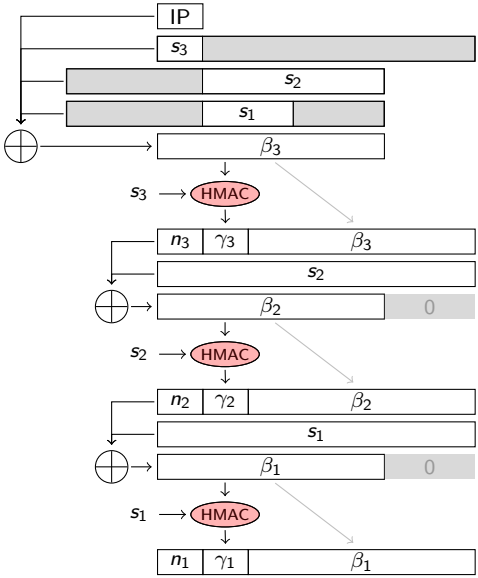
- Need **homomorphic properties** to split computation and aggregate results.
- Secure homomorphic hash **seems impractical** (no promising solutions found).
- Exploring **homomorphic encryption** as an alternative.

1. **RSA**
2. **ElGamal**

$$\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) = (g^{r_1}, m_1 \cdot h^{r_1})(g^{r_2}, m_2 \cdot h^{r_2})$$
$$= (g^{r_1+r_2}, (m_1 \cdot m_2)h^{r_1+r_2})$$
$$= \mathcal{E}(m_1 \cdot m_2)$$

3. **Paillier**

Limitation: Increase ciphertext size...



Adapting HMAC

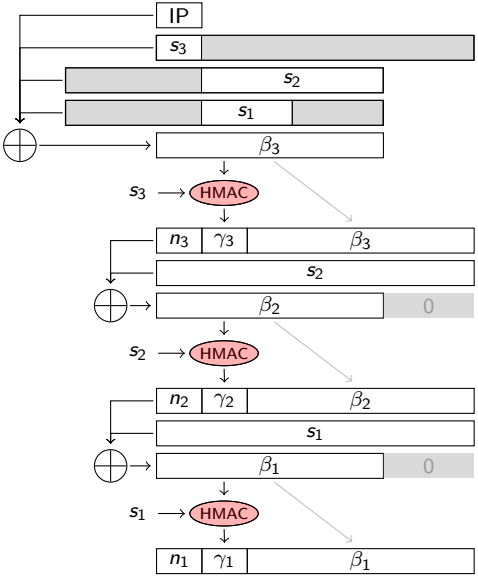
Main problem: Decentralizing a **Hash** ?

- Need **homomorphic properties** to split computation and aggregate results.
- Secure homomorphic hash **seems impractical** (no promising solutions found).
- Exploring **homomorphic encryption** as an alternative.

1. **RSA**

$$\mathcal{E}(m_1) \cdot \mathcal{E}(m_2) = m_1^e m_2^e \bmod n$$
$$= (m_1 m_2)^e \bmod n$$
$$= \mathcal{E}(m_1 \cdot m_2)$$

2. **ElGamal**
3. **Paillier**



Adapting HMAC

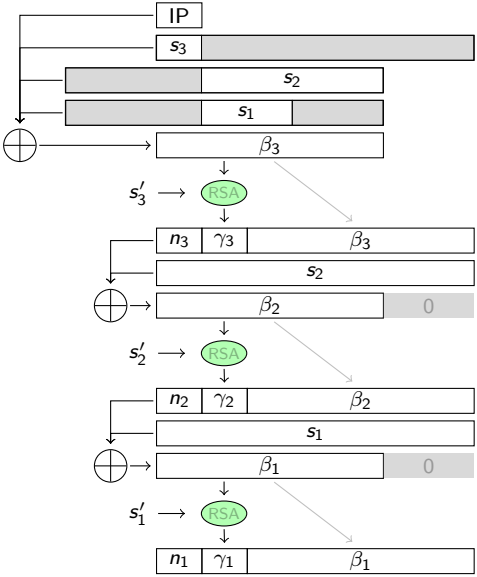
Main problem: Decentralizing a **Hash** ?

- Need **homomorphic properties** to split computation and aggregate results.
- Secure homomorphic hash **seems impractical** (no promising solutions found).
- Exploring **homomorphic encryption** as an alternative.

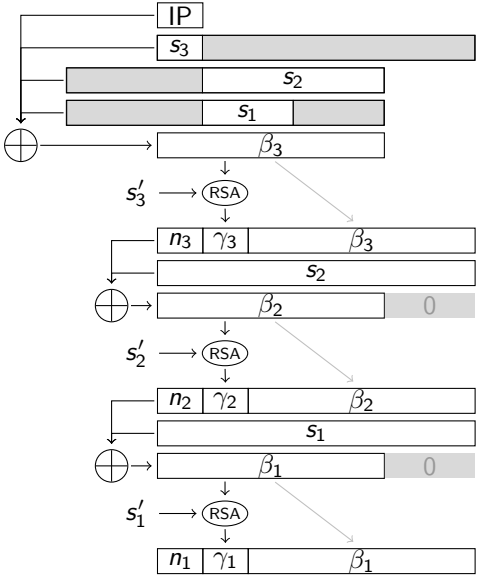
1. **RSA**
2. **ElGamal**
3. **Paillier**

Selected solution: RSA for integrity tag

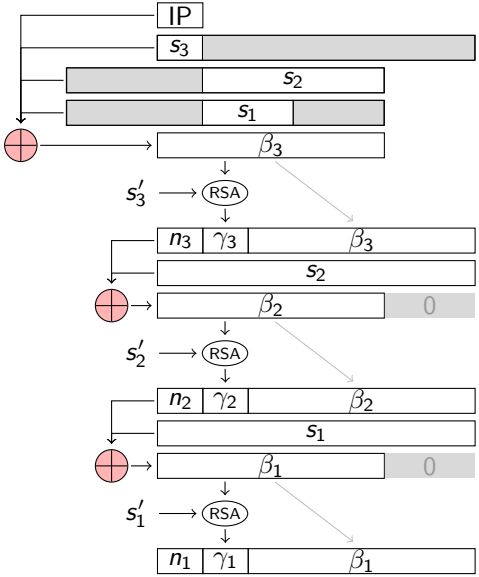
NB: s_i is different for each TTP but RSA required the same e...
Thus, create a new shared secret s'_i common to all TTP



Adapting XOR

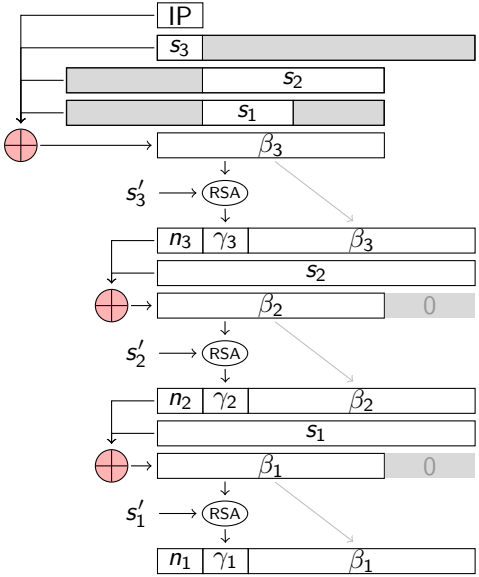


Adapting XOR



Adapting XOR

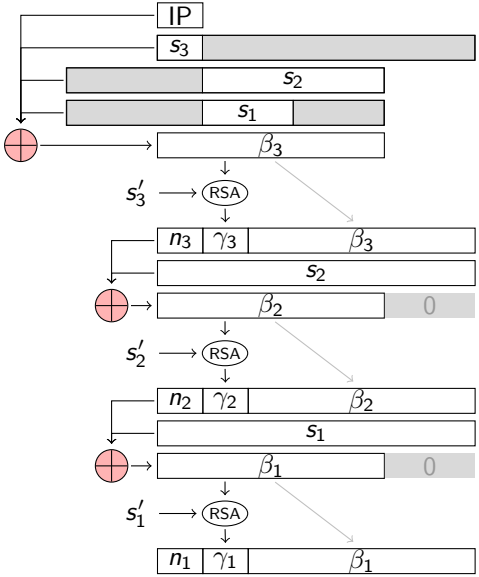
Since we use **RSA** for integrity tag γ_i



Adapting XOR

Since we use **RSA** for integrity tag γ_i

$$\begin{aligned}\mathcal{E}(m_1)\cdot\mathcal{E}(m_2) &= m_1^e m_2^e \bmod n \\ &= (m_1 m_2)^e \bmod n \\ &= \mathcal{E}(m_1 \cdot m_2)\end{aligned}$$

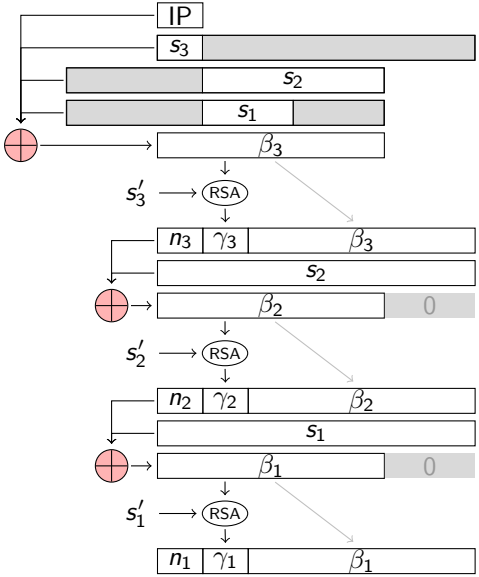


Adapting XOR

Since we use **RSA** for integrity tag γ_i

$$\begin{aligned}\mathcal{E}(m_1)\cdot\mathcal{E}(m_2) &= m_1^e m_2^e \bmod n \\ &= (m_1 m_2)^e \bmod n \\ &= \mathcal{E}(m_1 \cdot m_2)\end{aligned}$$

Modular multiplication of integrity tags gives integrity tag of headers' modular product.



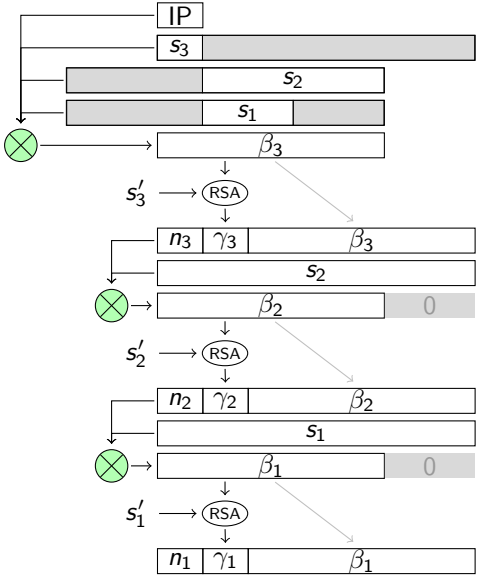
Adapting XOR

Since we use **RSA** for integrity tag γ_i

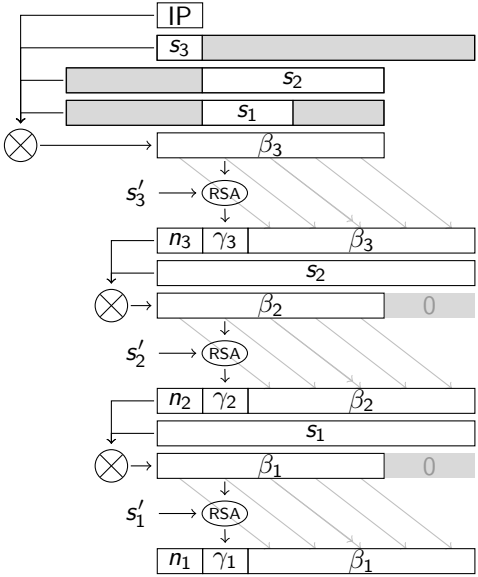
$$\begin{aligned}\mathcal{E}(m_1)\cdot\mathcal{E}(m_2) &= m_1^e m_2^e \bmod n \\ &= (m_1 m_2)^e \bmod n \\ &= \mathcal{E}(m_1 \cdot m_2)\end{aligned}$$

Modular multiplication of integrity tags gives integrity tag of headers' modular product.

Thus, header elements must be combined via modular multiplication rather than XOR.

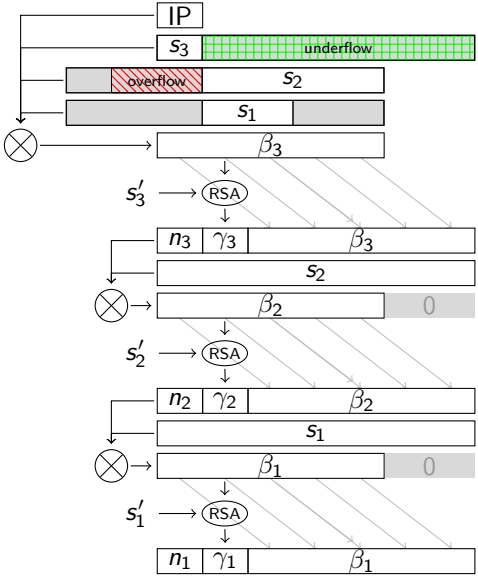


Handle overflows



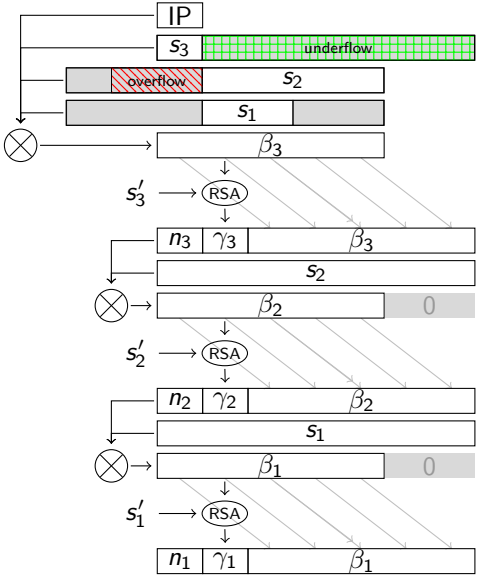
Handle overflows

- **Current Challenge: Overflow** issues



Handle overflows

- **Current Challenge: Overflow** issues
- Handling these issues is challenging. It may lead to **information loss** (further research is required).



Handle overflows

- **Current Challenge: Overflow** issues
- Handling these issues is challenging. It may lead to **information loss** (further research is required).
- **Proposed solution:** Simplify by **dividing data into small chunks** and processing each chunk modulo its size.