unipampa
Universidade Federal do Pampa

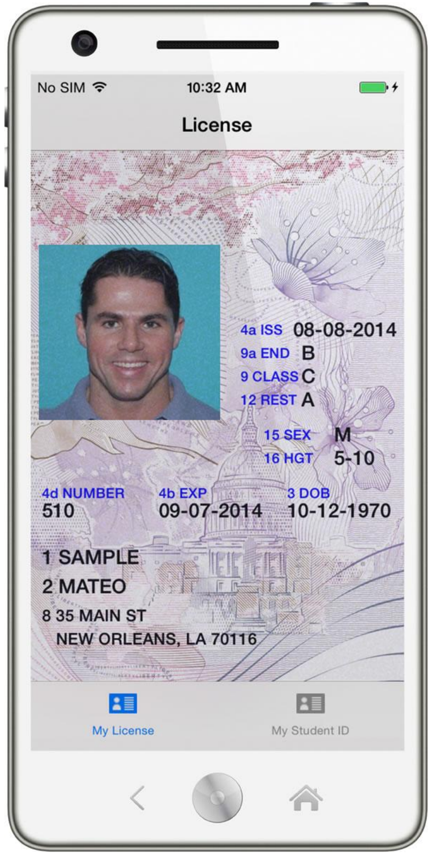
UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

UDESC

Auth4App: Protocols for Identification and Authentication using Mobile Applications

Diego Kreutz, Rafael Fernandes, Giulliano Paz, Tadeu Jenuario, Rodrigo Mansilha, Roger Immich, Charles C. Miers

Aplicativos de Identificação



License

4a ISS 08-08-2014
9a END B
9 CLASS C
12 REST A
15 SEX M
16 HGT 5-10

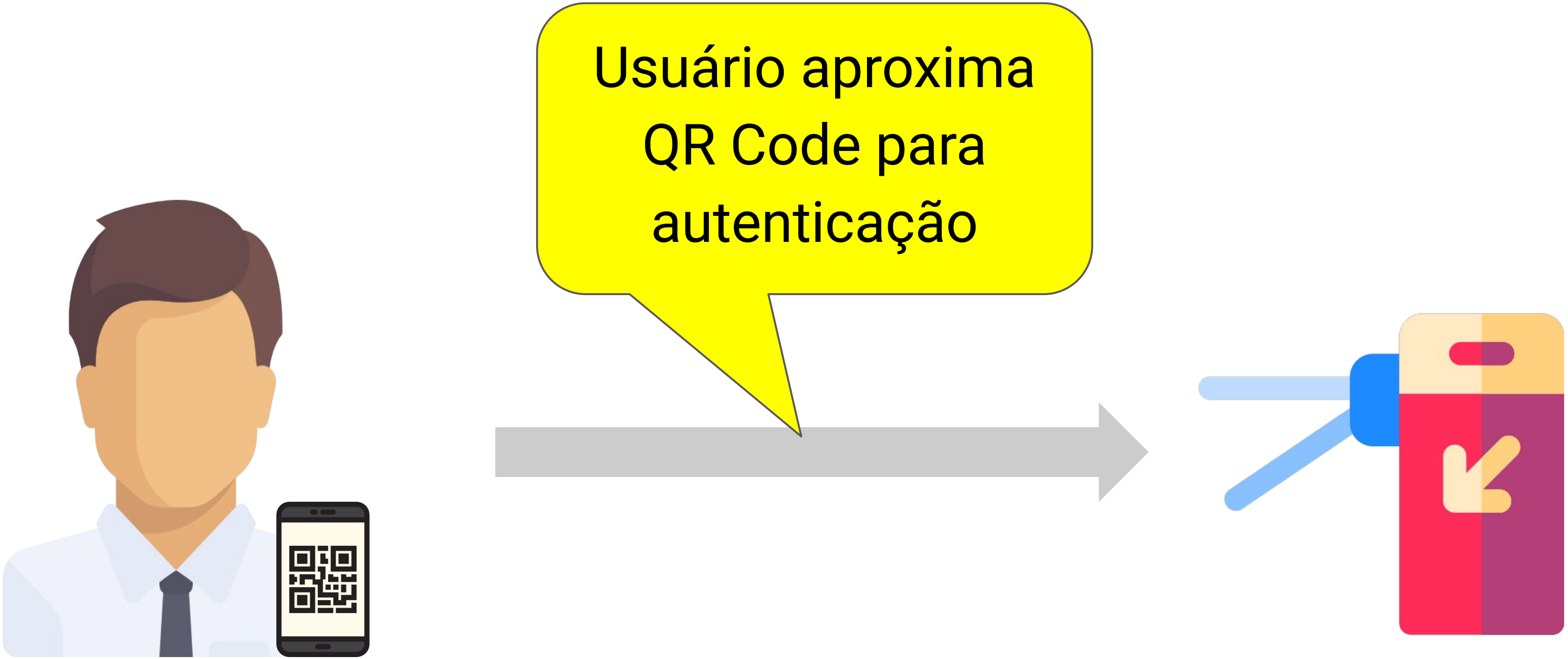
4d NUMBER 510
4b EXP 09-07-2014
3 DOB 10-12-1970

1 SAMPLE
2 MATEO
8 35 MAIN ST
NEW ORLEANS, LA 70116

Aplicativos e Mecanismos de Verificação

VALID
snrgé
Shunji Nishzawa

Estudo de Caso: SESC-RS

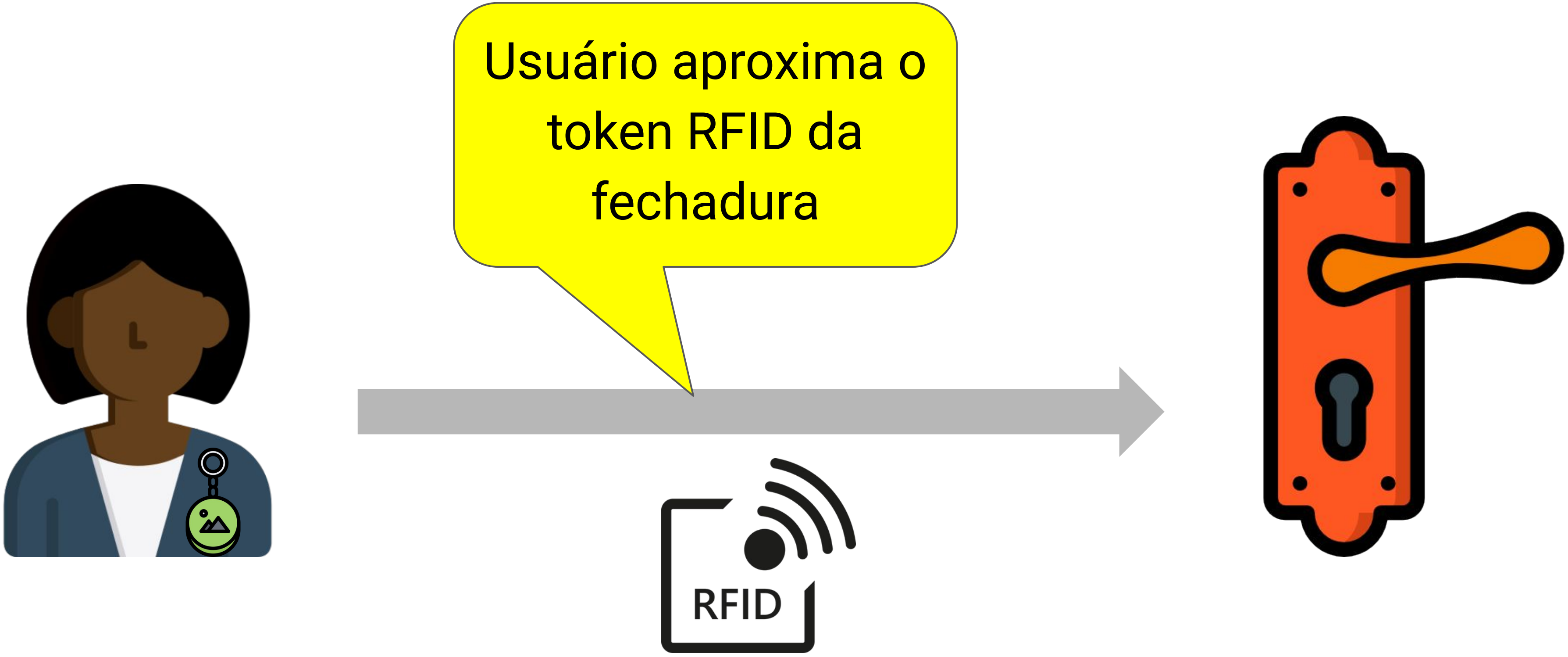


Usuário aproxima QR Code para autenticação

Estudo de Caso: SESC-RS

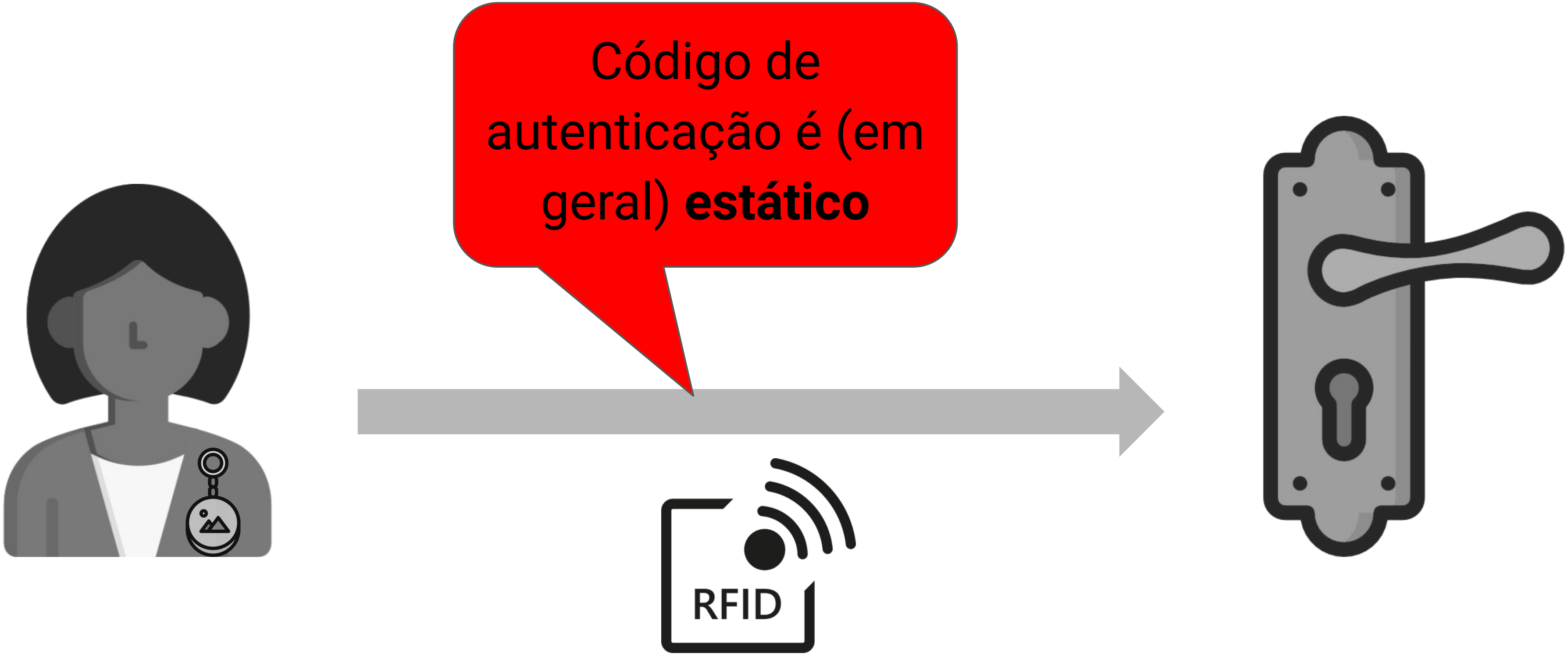
Código de Autenticação (QR Code) **estático**

Automação e Segurança Residencial



Usuário aproxima o token RFID da fechadura

Automação e Segurança Residencial



Código de autenticação é (em geral) **estático**

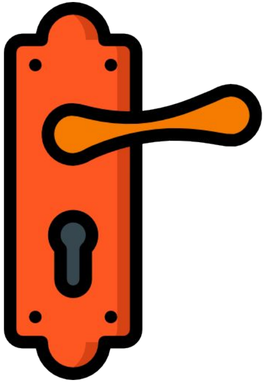
O problema: códigos estáticos

Códigos de autenticação **estáticos**

Objetivo: códigos dinâmicos

Códigos **robustos, únicos** e **descartáveis**

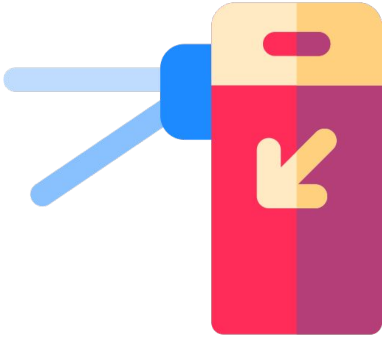
Desafio: segurança e usabilidade



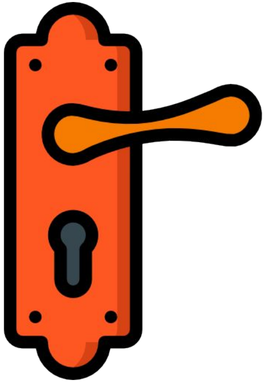
Posse

+

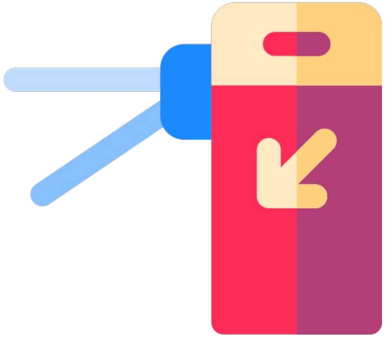
“Identidade”



Desafio: posse



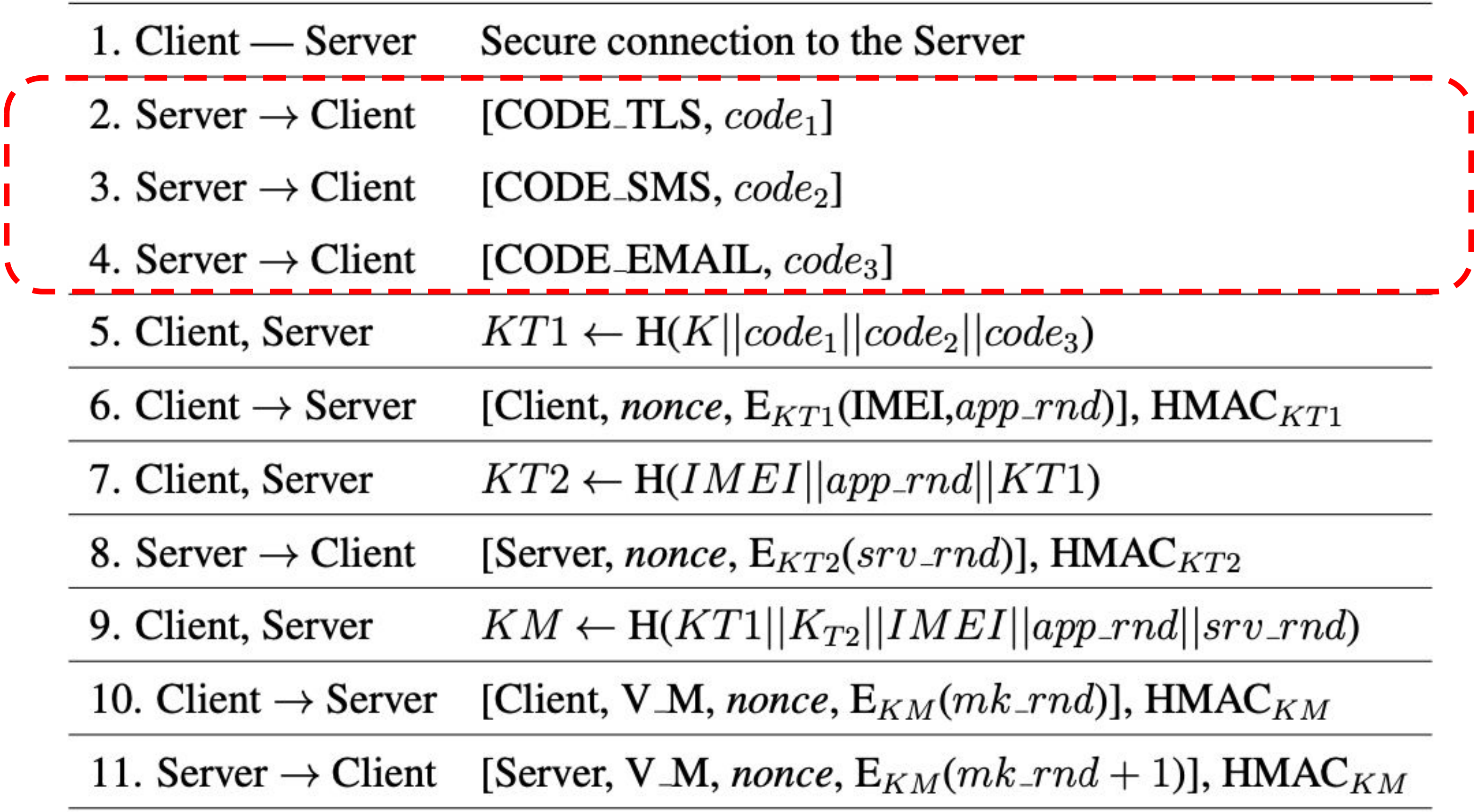
Posse requer vínculo forte com o usuário



Protocolo de identificação e vínculo

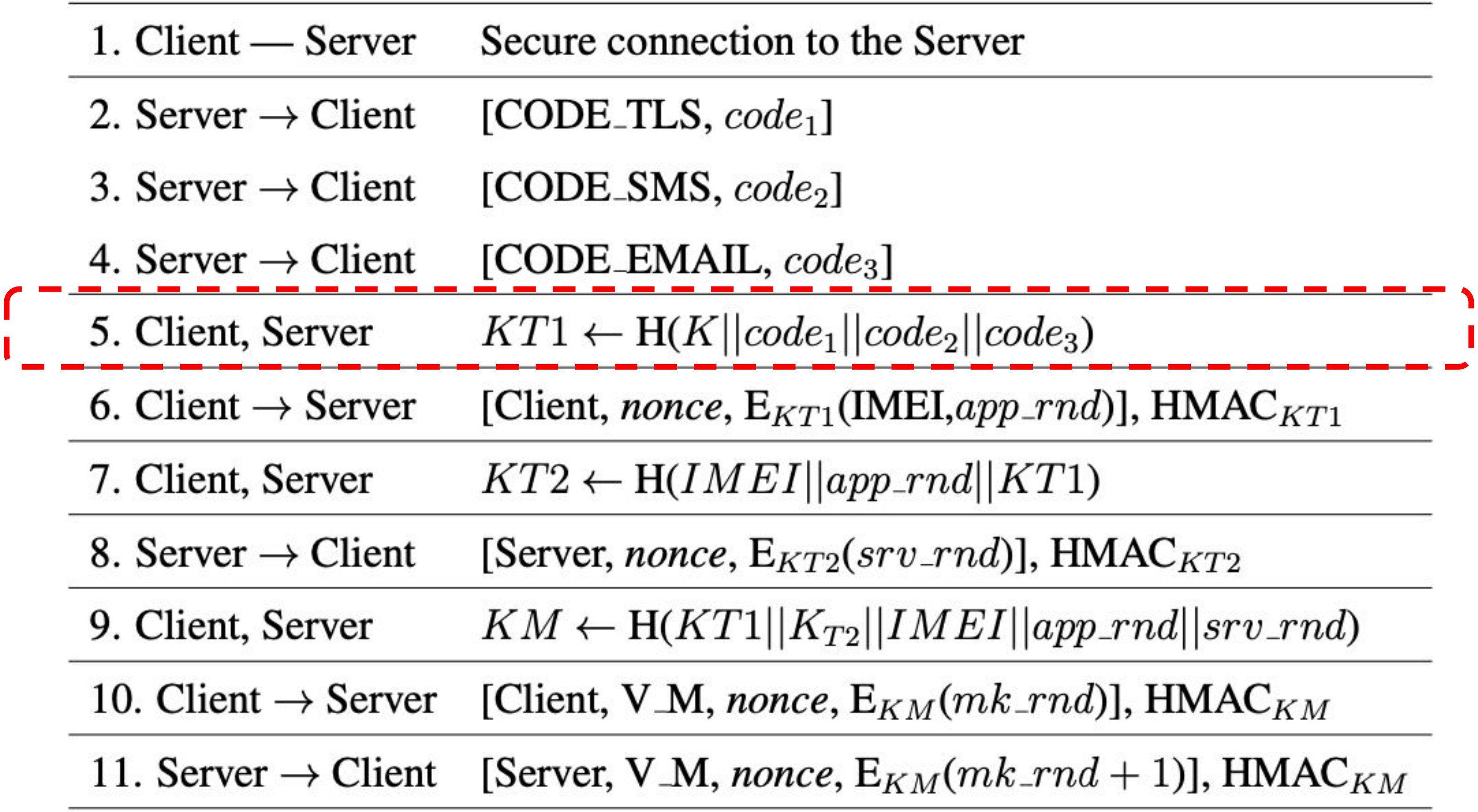
- Identificar usuários
- Vincular identidade ao aplicativo e dispositivo
- Gerar chave mestra

Protocolo de identificação e vínculo



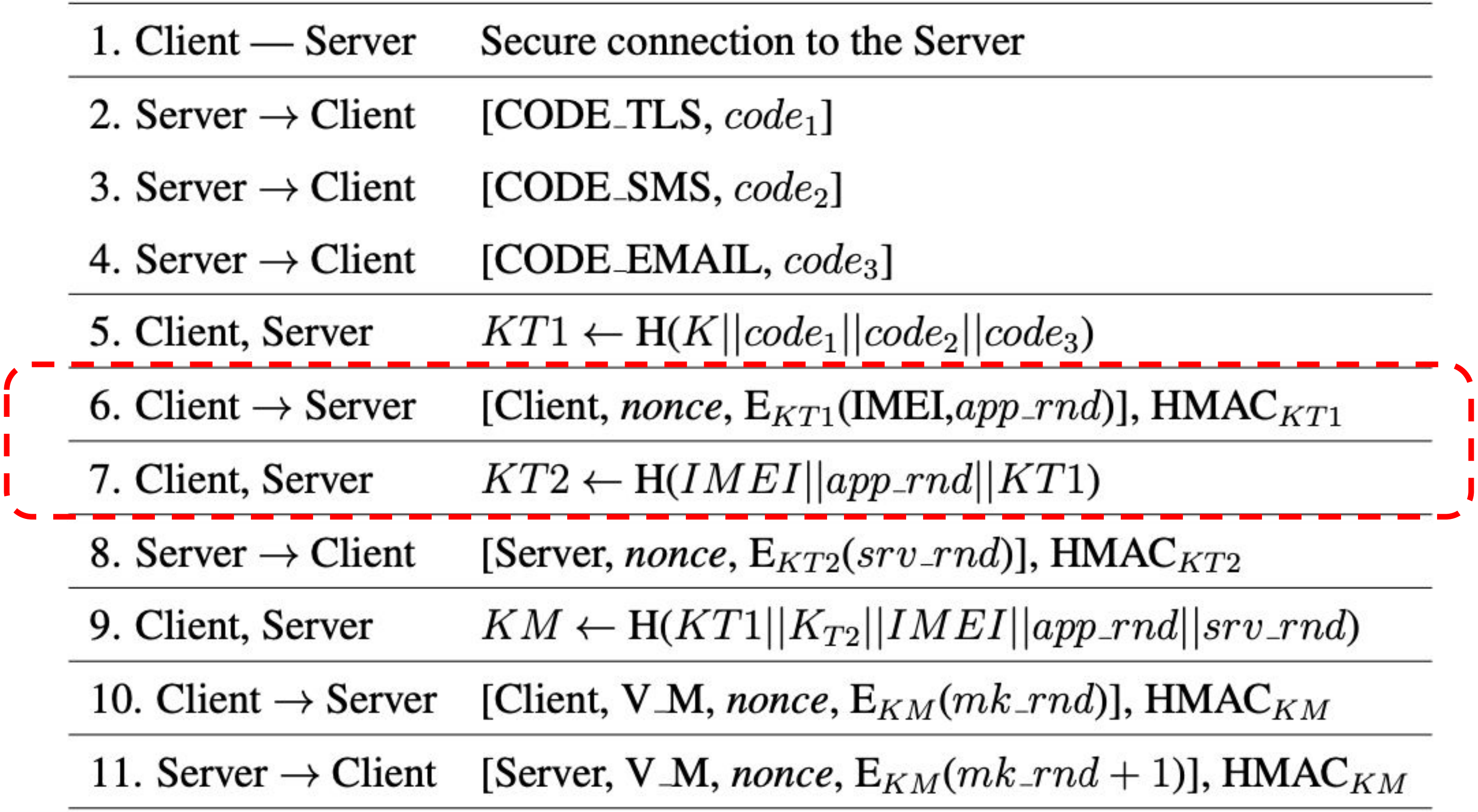
1. Client — Server	Secure connection to the Server
2. Server → Client	[CODE_TLS, $code_1$]
3. Server → Client	[CODE_SMS, $code_2$]
4. Server → Client	[CODE_EMAIL, $code_3$]
5. Client, Server	$KT1 \leftarrow \mathrm{H}(K\|\|code_1\|\|code_2\|\|code_3)$
6. Client → Server	[Client, $nonce$, E_{KT1}(IMEI,app_rnd)], HMAC_{KT1}
7. Client, Server	$KT2 \leftarrow \mathrm{H}(IMEI\|\|app\_rnd\|\|KT1)$
8. Server → Client	[Server, $nonce$, $\mathrm{E}_{KT2}(srv\_rnd)$], HMAC_{KT2}
9. Client, Server	$KM \leftarrow \mathrm{H}(KT1\|\|K_{T2}\|\|IMEI\|\|app\_rnd\|\|srv\_rnd)$
10. Client → Server	[Client, V_M, $nonce$, $\mathrm{E}_{KM}(mk\_rnd)$], HMAC_{KM}
11. Server → Client	[Server, V_M, $nonce$, $\mathrm{E}_{KM}(mk\_rnd + 1)$], HMAC_{KM}

Protocolo de identificação e vínculo



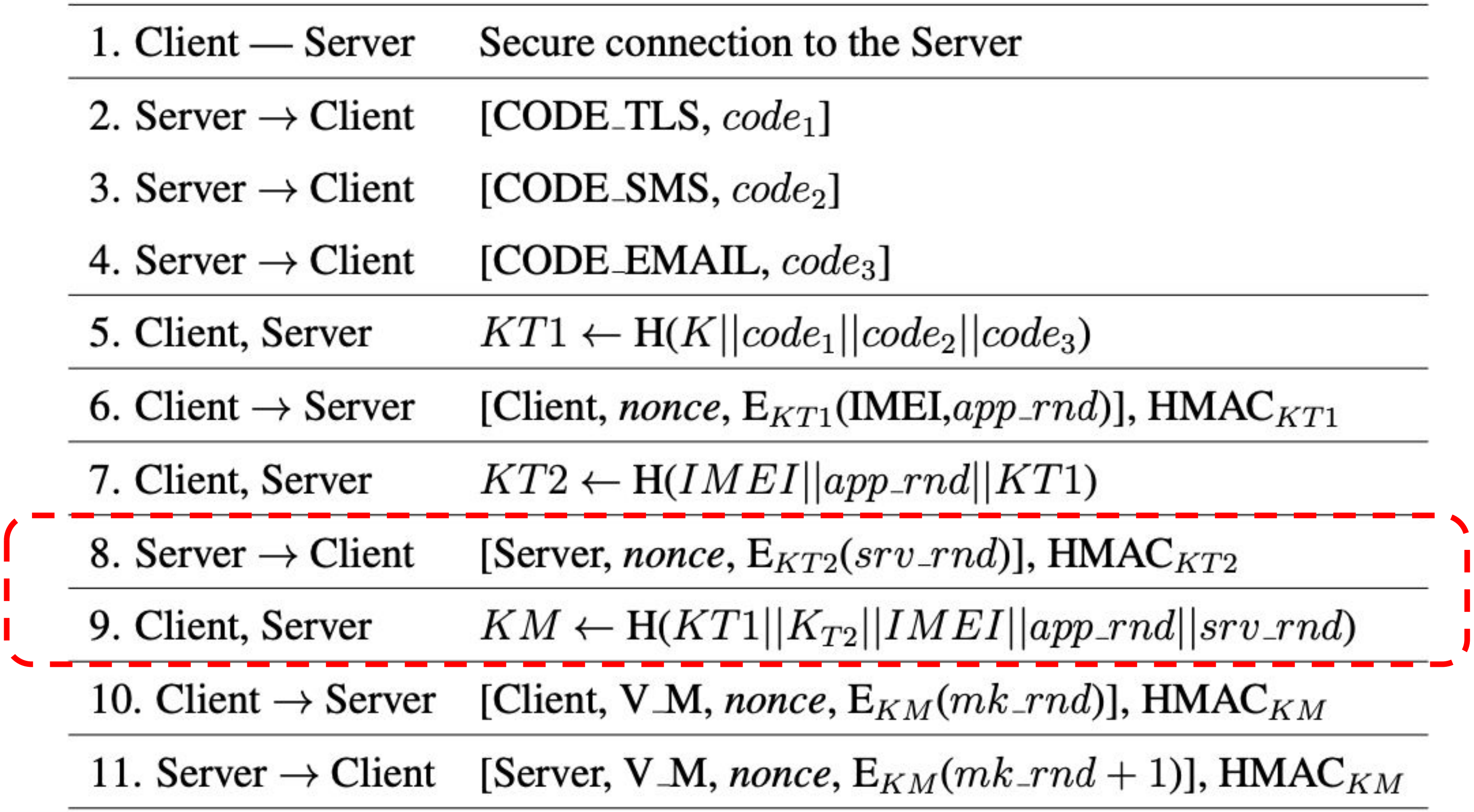
1. Client — Server	Secure connection to the Server
2. Server → Client	[CODE_TLS, $code_1$]
3. Server → Client	[CODE_SMS, $code_2$]
4. Server → Client	[CODE_EMAIL, $code_3$]
5. Client, Server	$KT1 \leftarrow \mathrm{H}(K \Vert code_1 \Vert code_2 \Vert code_3)$
6. Client → Server	[Client, *nonce*, E_{KT1}(IMEI,app_rnd)], HMAC_{KT1}
7. Client, Server	$KT2 \leftarrow \mathrm{H}(IMEI \Vert app\_rnd \Vert KT1)$
8. Server → Client	[Server, *nonce*, $\mathrm{E}_{KT2}(srv\_rnd)$], HMAC_{KT2}
9. Client, Server	$KM \leftarrow \mathrm{H}(KT1 \Vert K_{T2} \Vert IMEI \Vert app\_rnd \Vert srv\_rnd)$
10. Client → Server	[Client, V_M, *nonce*, $\mathrm{E}_{KM}(mk\_rnd)$], HMAC_{KM}
11. Server → Client	[Server, V_M, *nonce*, $\mathrm{E}_{KM}(mk\_rnd + 1)$], HMAC_{KM}

Protocolo de identificação e vínculo



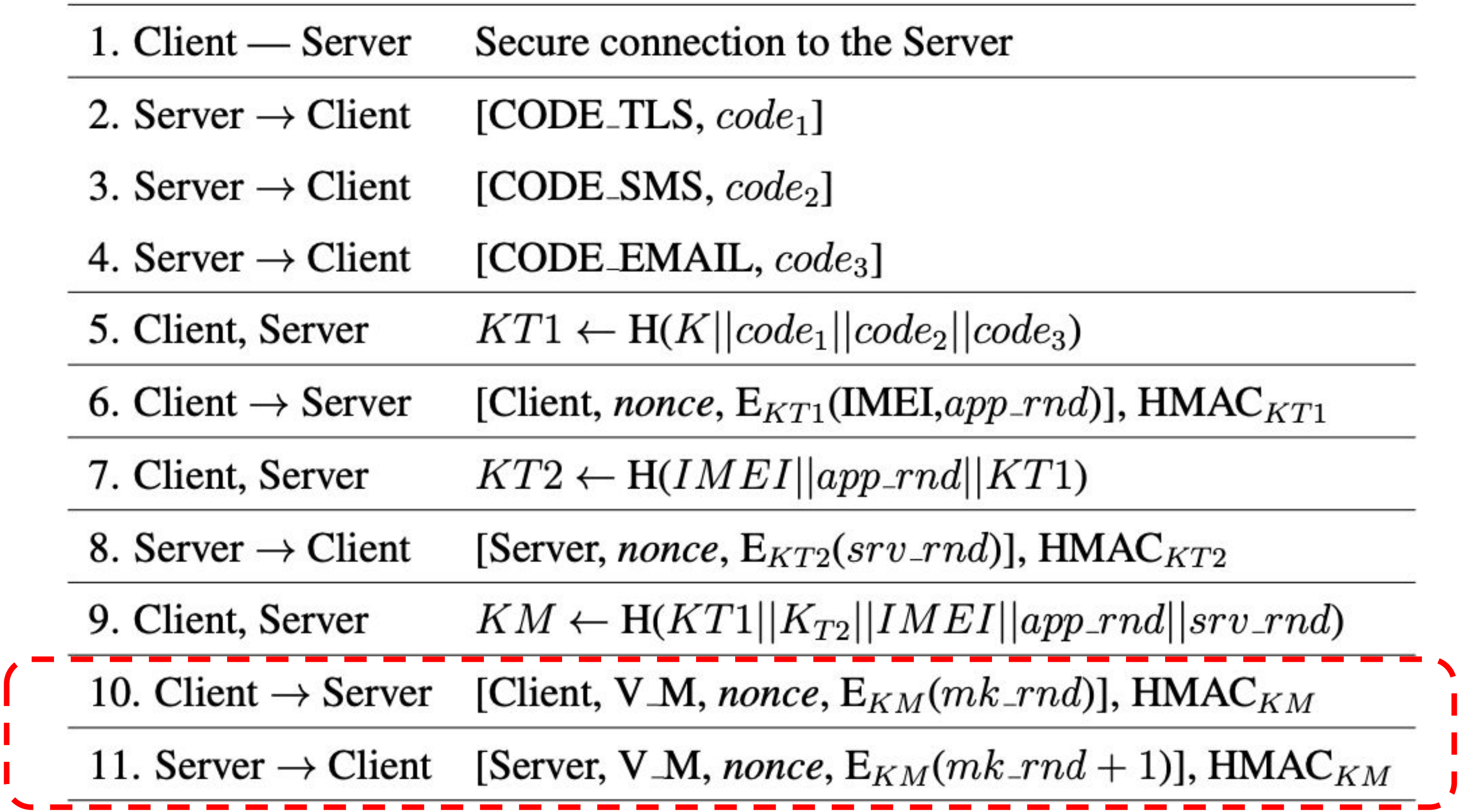
1. Client — Server	Secure connection to the Server
2. Server → Client	[CODE_TLS, $code_1$]
3. Server → Client	[CODE_SMS, $code_2$]
4. Server → Client	[CODE_EMAIL, $code_3$]
5. Client, Server	$KT1 \leftarrow \mathrm{H}(K \| code_1 \| code_2 \| code_3)$
6. Client → Server	[Client, *nonce*, E_{KT1}(IMEI,app_rnd)], HMAC_{KT1}
7. Client, Server	$KT2 \leftarrow \mathrm{H}(IMEI \| app\_rnd \| KT1)$
8. Server → Client	[Server, *nonce*, $\mathrm{E}_{KT2}(srv\_rnd)$], HMAC_{KT2}
9. Client, Server	$KM \leftarrow \mathrm{H}(KT1 \| K_{T2} \| IMEI \| app\_rnd \| srv\_rnd)$
10. Client → Server	[Client, V_M, *nonce*, $\mathrm{E}_{KM}(mk\_rnd)$], HMAC_{KM}
11. Server → Client	[Server, V_M, *nonce*, $\mathrm{E}_{KM}(mk\_rnd + 1)$], HMAC_{KM}

Protocolo de identificação e vínculo



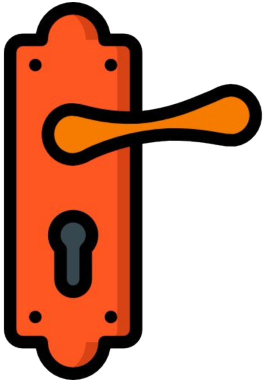
1. Client — Server	Secure connection to the Server
2. Server → Client	[CODE_TLS, $code_1$]
3. Server → Client	[CODE_SMS, $code_2$]
4. Server → Client	[CODE_EMAIL, $code_3$]
5. Client, Server	$KT1 \leftarrow \mathrm{H}(K\|\|code_1\|\|code_2\|\|code_3)$
6. Client → Server	[Client, *nonce*, E_{KT1}(IMEI,app_rnd)], HMAC_{KT1}
7. Client, Server	$KT2 \leftarrow \mathrm{H}(IMEI\|\|app\_rnd\|\|KT1)$
8. Server → Client	[Server, *nonce*, $\mathrm{E}_{KT2}(srv\_rnd)$], HMAC_{KT2}
9. Client, Server	$KM \leftarrow \mathrm{H}(KT1\|\|K_{T2}\|\|IMEI\|\|app\_rnd\|\|srv\_rnd)$
10. Client → Server	[Client, V_M, *nonce*, $\mathrm{E}_{KM}(mk\_rnd)$], HMAC_{KM}
11. Server → Client	[Server, V_M, *nonce*, $\mathrm{E}_{KM}(mk\_rnd + 1)$], HMAC_{KM}

Protocolo de identificação e vínculo

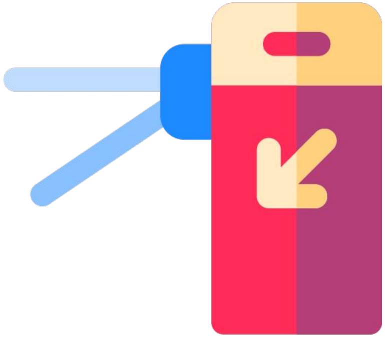


1. Client — Server	Secure connection to the Server
2. Server → Client	[CODE_TLS, $code_1$]
3. Server → Client	[CODE_SMS, $code_2$]
4. Server → Client	[CODE_EMAIL, $code_3$]
5. Client, Server	$KT1 \leftarrow$ H($K \vert\vert code_1 \vert\vert code_2 \vert\vert code_3$)
6. Client → Server	[Client, *nonce*, E_{KT1}(IMEI,app_rnd)], HMAC_{KT1}
7. Client, Server	$KT2 \leftarrow$ H($IMEI \vert\vert app\_rnd \vert\vert KT1$)
8. Server → Client	[Server, *nonce*, E_{KT2}(srv_rnd)], HMAC_{KT2}
9. Client, Server	$KM \leftarrow$ H($KT1 \vert\vert K_{T2} \vert\vert IMEI \vert\vert app\_rnd \vert\vert srv\_rnd$)
10. Client → Server	[Client, V_M, *nonce*, E_{KM}(mk_rnd)], HMAC_{KM}
11. Server → Client	[Server, V_M, *nonce*, E_{KM}($mk\_rnd + 1$)], HMAC_{KM}

Desafio: “identidade”



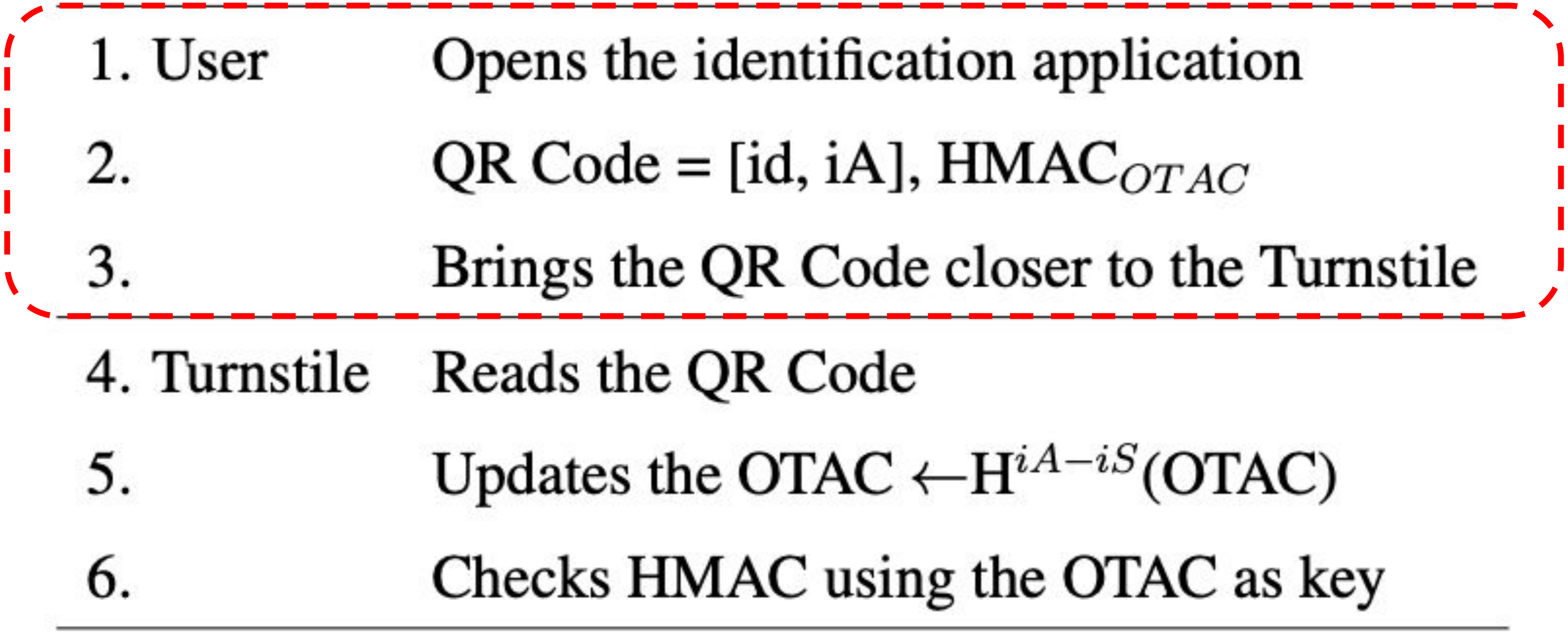
“Identidade”
requer protocolo
de verificação



Protocolo de autenticação/verificação

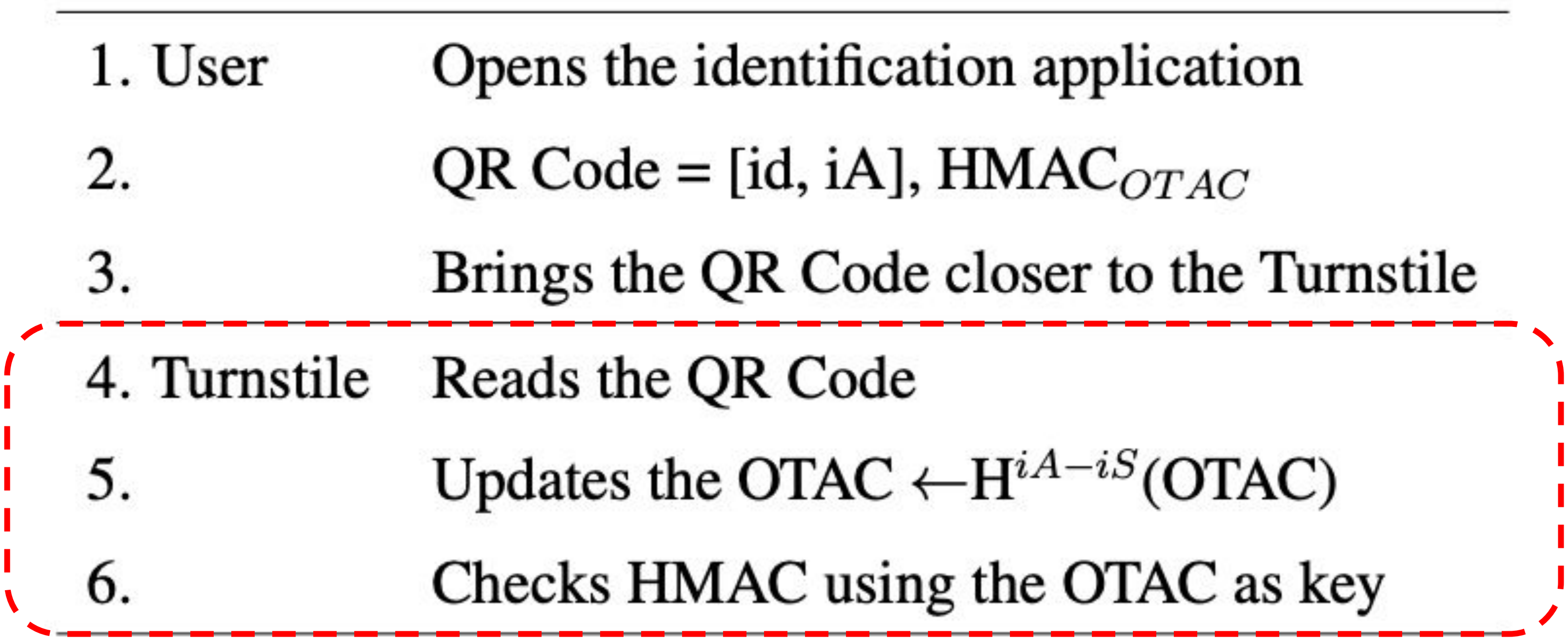
- Esquema de autenticação simples
- Gerador de códigos dinâmicos e únicos
- Inicialização utilizando a chave mestra

Protocolo de autenticação/verificação



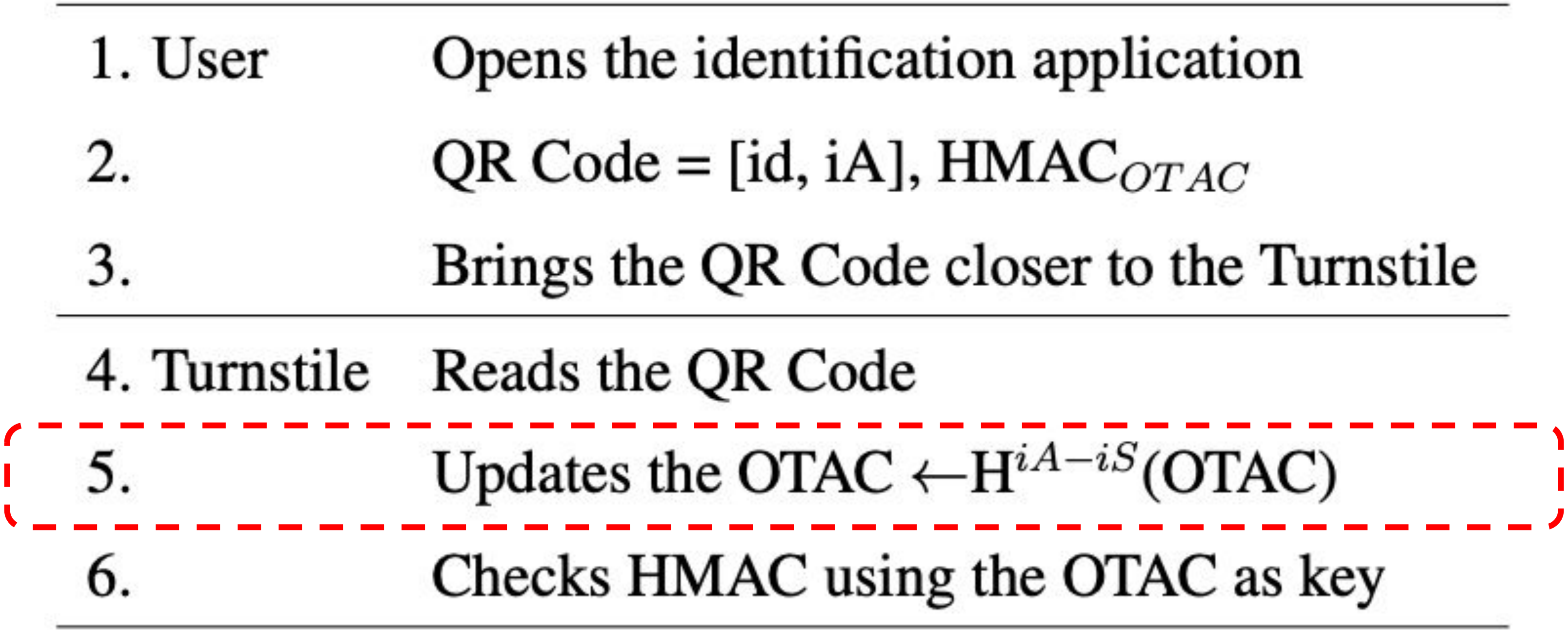
1.	User	Opens the identification application
2.		QR Code = [id, iA], HMAC$_{OTAC}$
3.		Brings the QR Code closer to the Turnstile
4.	Turnstile	Reads the QR Code
5.		Updates the OTAC $\leftarrow$H^{iA-iS}(OTAC)
6.		Checks HMAC using the OTAC as key

Protocolo de autenticação/verificação



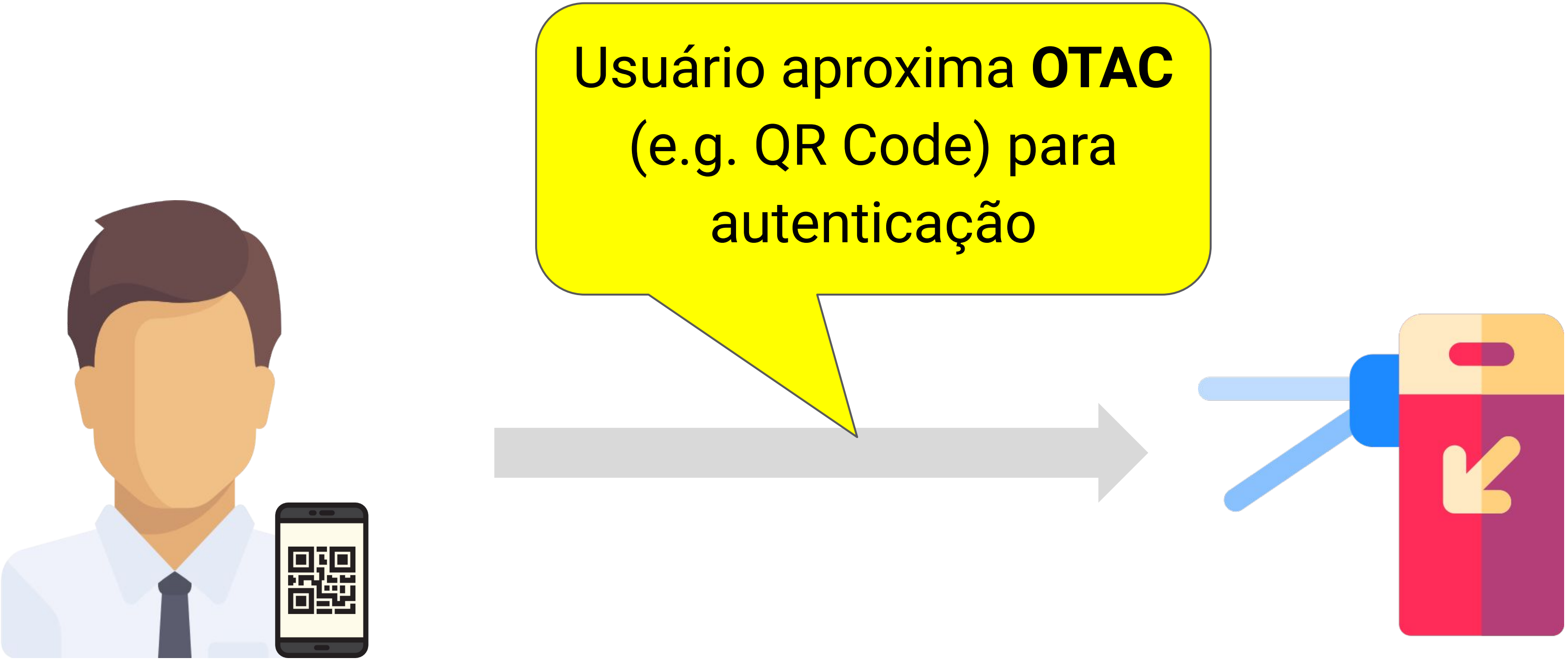
1. User	Opens the identification application
2.	QR Code = [id, iA], HMAC_{OTAC}
3.	Brings the QR Code closer to the Turnstile
4. Turnstile	Reads the QR Code
5.	Updates the OTAC $\leftarrow \text{H}^{iA-iS}(\text{OTAC})$
6.	Checks HMAC using the OTAC as key

Protocolo de autenticação/verificação



1. User	Opens the identification application
2.	QR Code = [id, iA], HMAC$_{OTAC}$
3.	Brings the QR Code closer to the Turnstile
4. Turnstile	Reads the QR Code
5.	Updates the OTAC $\leftarrow$H^{iA-iS}(OTAC)
6.	Checks HMAC using the OTAC as key

Caso de uso: catracas eletrônicas

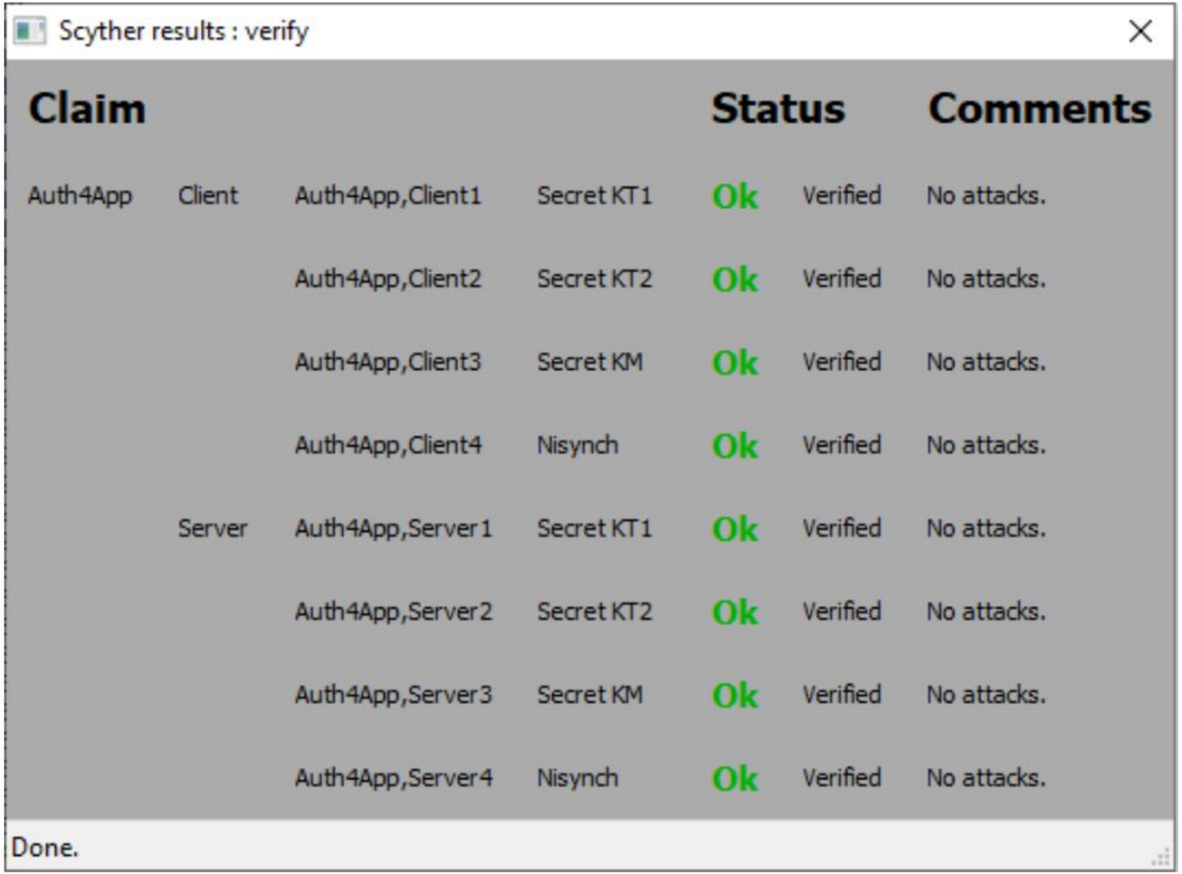


Usuário aproxima **OTAC** (e.g. QR Code) para autenticação

Caso de uso: fechaduras inteligentes

Usuário aproxima smartphone (e.g. **NFC + OTAC**) da fechadura

Verificação automática com Scyther

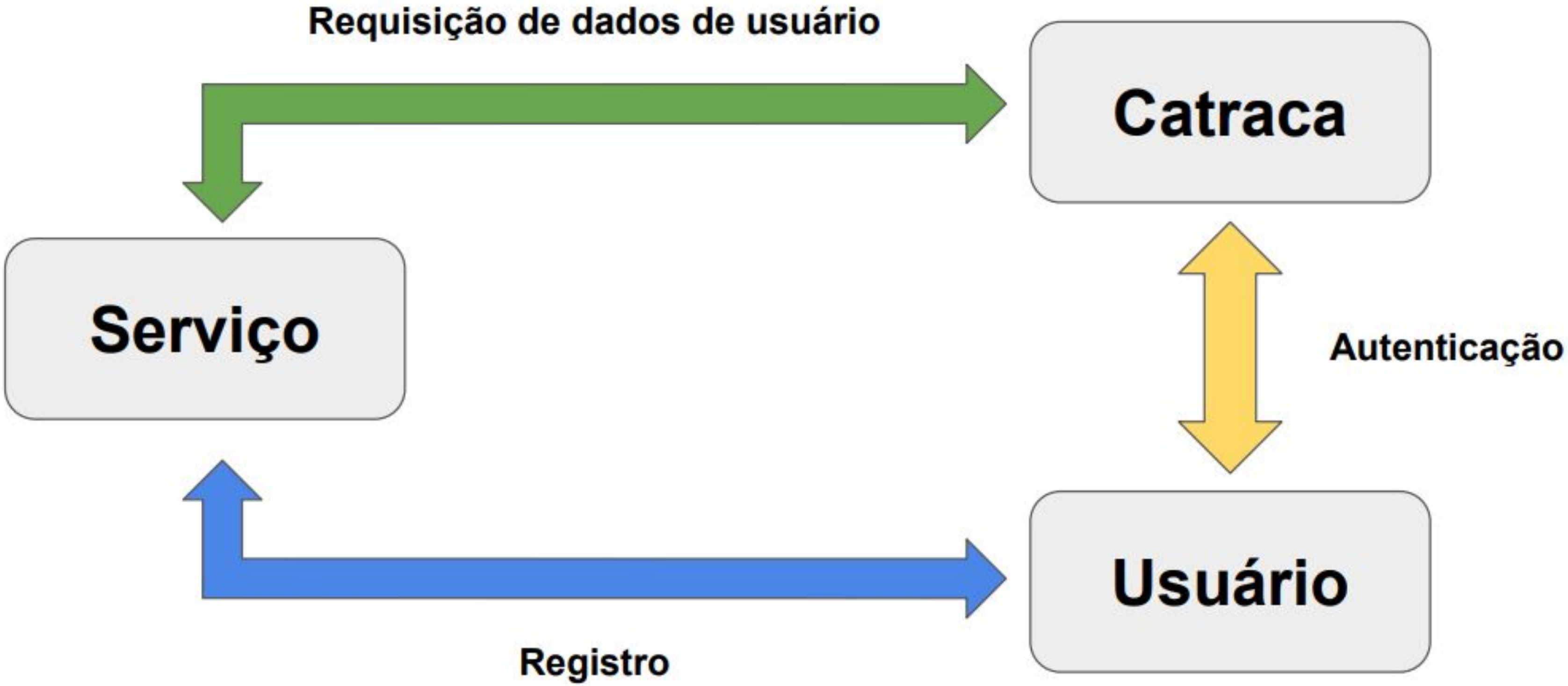


Scyther results : verify

Claim				Status		Comments
Auth4App	Client	Auth4App,Client1	Secret KT1	Ok	Verified	No attacks.
		Auth4App,Client2	Secret KT2	Ok	Verified	No attacks.
		Auth4App,Client3	Secret KM	Ok	Verified	No attacks.
		Auth4App,Client4	Nisynch	Ok	Verified	No attacks.
	Server	Auth4App,Server1	Secret KT1	Ok	Verified	No attacks.
		Auth4App,Server2	Secret KT2	Ok	Verified	No attacks.
		Auth4App,Server3	Secret KM	Ok	Verified	No attacks.
		Auth4App,Server4	Nisynch	Ok	Verified	No attacks.

Done.

Experimentos - ambiente



Experimentos - resultados

- Geração de QR Code = 9,17ms
- Leitura e manipulação de QR Code = 5,34ms
- Verificação do OTAC = 0,04ms
- Calcular um OTAC = 0,03ms

Trabalhos futuros

Hardware-assisted security com TEEs

Modelagem de ataques sofisticados

Verificação formal (e.g., Tamarin)

Gerenciamento de provas formais (e.g., Coq)