О криптографических механизмах и протоколах существующих защищенных мессенджеров

Николаев В.Д., Ахметзянова Л.Р.

МГУ им. М.В. Ломоносова, ООО «КРИПТО-ПРО»

Мессенджеры — самые часто используемые приложения для смартфонов

Telegram, Skype, WhatsApp, Viber, Google Hangouts, Wickr, Signal, TextSecure, ICQ, Silent Phone, CryptoChat, Threema, Chadder, RedPhone, ProtonMail, Facebook Messenger, Google Allo, . . .

Что понимается под «конфиденциальностью» пользовательской информации?

Централизованное управление: между пользователями устанавливается защищенное соединение через внешний сервер с использованием безопасных транспортных протоколов

Ключи для расшифрования сообщений пользователей хранятся на внешнем сервере ⇒ сервер может получить доступ к переписке.

Доверяем ли мы серверу?

End-to-End шифрование (E2E): используется криптографический протокол, посредством которого устанавливается защищенное соединение непосредственно между пользователями

Ключи для расшифрования сообщений хранятся на устройствах пользователей, а не на внешних серверах $\Rightarrow$ никто, кроме пользователей, не может получить доступ к переписке.

Как удостовериться, что мессенджер действительно обеспечивает заявленный уровень защищенности пользовательской информации?

Стойкий E2E протокол

+

«Безопасная» программная реализация

Стойкий E2E протокол: формальное верифицированное(!) обоснование стойкости протокола в приближенной к реальности модели противника.

«Безопасная» программная реализация: open-source code, для которого на данный момент не известно никаких эксплуатируемых уязвимостей.

CRYPTOPRO

Telegram vs WhatsApp vs Signal vs Wickr

- Обоснование стойкости используемого E2E протокола;
- Аутентификация;
- Базовые криптографические протоколы и примитивы;
- Побочная информация (metadata) на сервере;
- Хранение конфиденциальных данных на конечных устройствах;
- Известные уязвимости в протоколе и/или реализации.

CRYPTOPRO

Протокол: Signal
Разработчик: Open Whisper Systems
Прототип: TextSecure
Реализация: WhatsApp, Signal, Facebook Messenger, Google Allo

- Регистрация пользователей на сервере (эллиптические кривые, подпись Ed25519)
- Инициализация долговременной защищенной сессии путем согласования «начального» симметричного ключа (triple-DH)
- Непосредственный обмен сообщениями с использованием техники «symmetric-ratcheting» (Encrypt-then-MAC с использованием AES in CBC mode и HMAC-SHA256, KDF)
- Пересогласование «начального» симметричного ключа (техника «assymmetric-ratcheting») (DH, KDF)

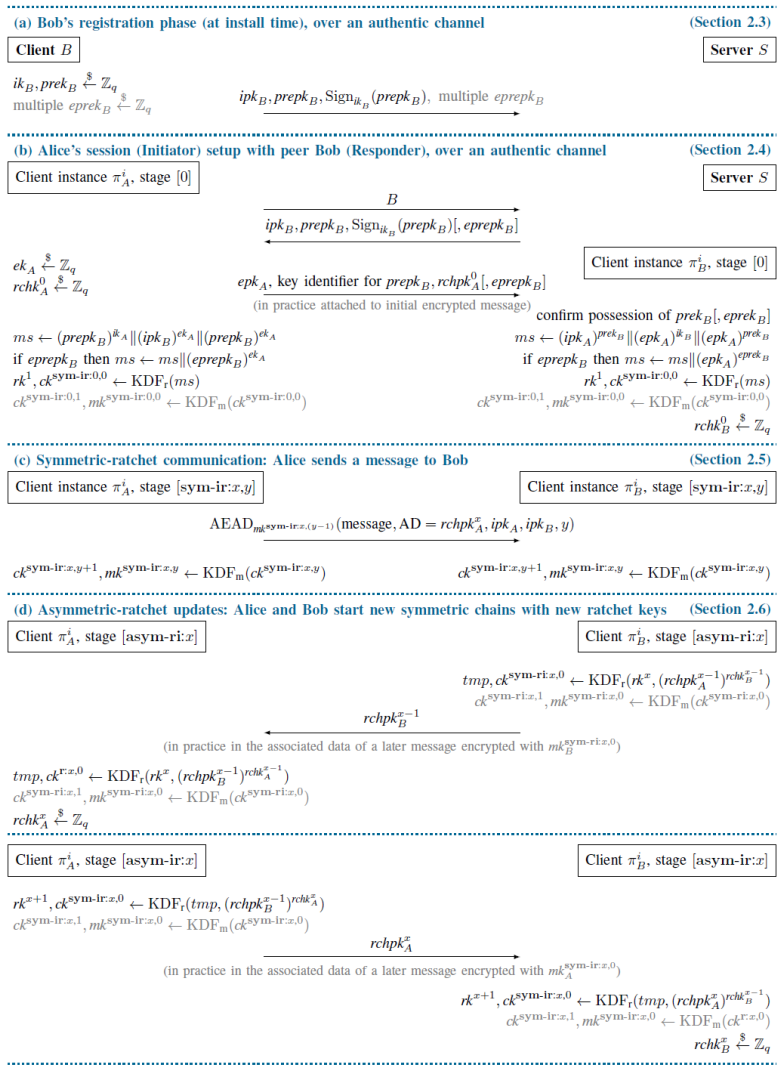
CRYPTOPRO

Статья «A Formal Security Analysis of the Signal Messaging Protocol»

Авторы: K. Cohn-Gordon
C. Cremers, B. Dowling
L. Garratt, D. Stebila

University of Oxford, UK
Queensland University of Technology, Australia
McMaster University, Canada

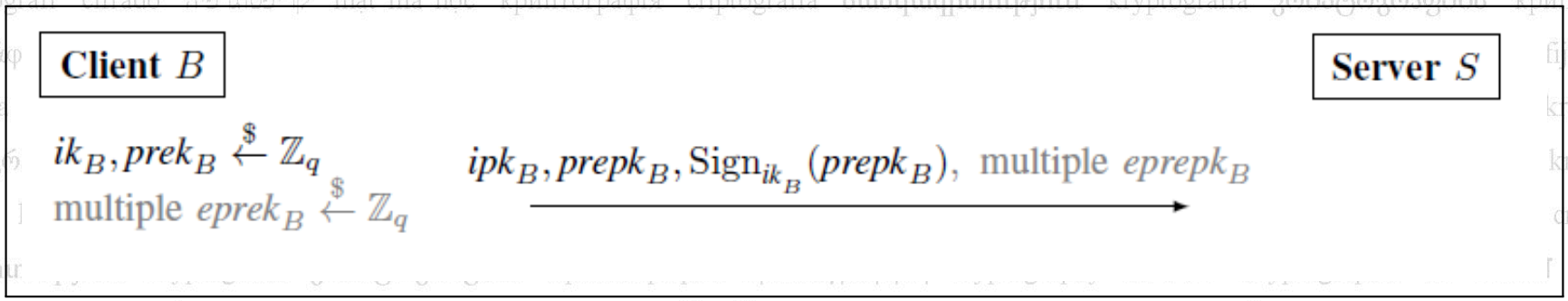
.../eprint.iacr.org/2016/1013.pdf



Установка: генерация ключевых пар

- (ik, ipk) (Identity Key Pair) — долговременный ключ;
- $(prek, prepk)$ (Signed Pre Key) — промежуточный ключ, где открытая часть $prepk$ подписана с помощью ключа ik (multiple peers);
- $\{(eprek, eprepk)\}$ (Ephemeral One-time Pre Keys) — набор одноразовых ключей (при необходимости пополняется).

Регистрация:

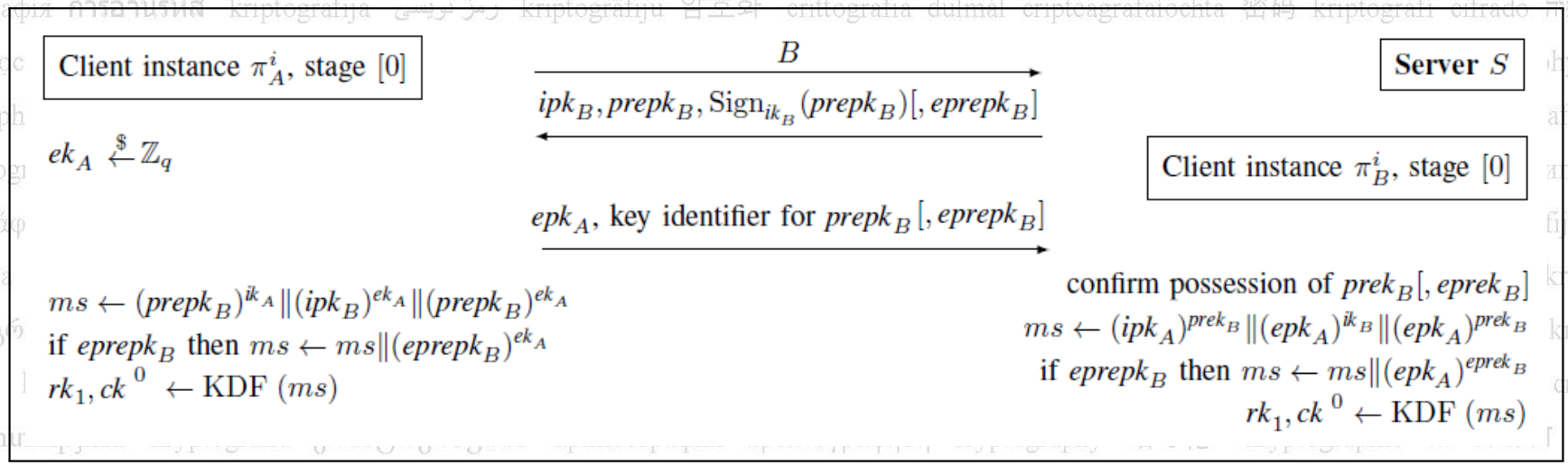


! Канал аутентифицированный !

CRYPTOPRO

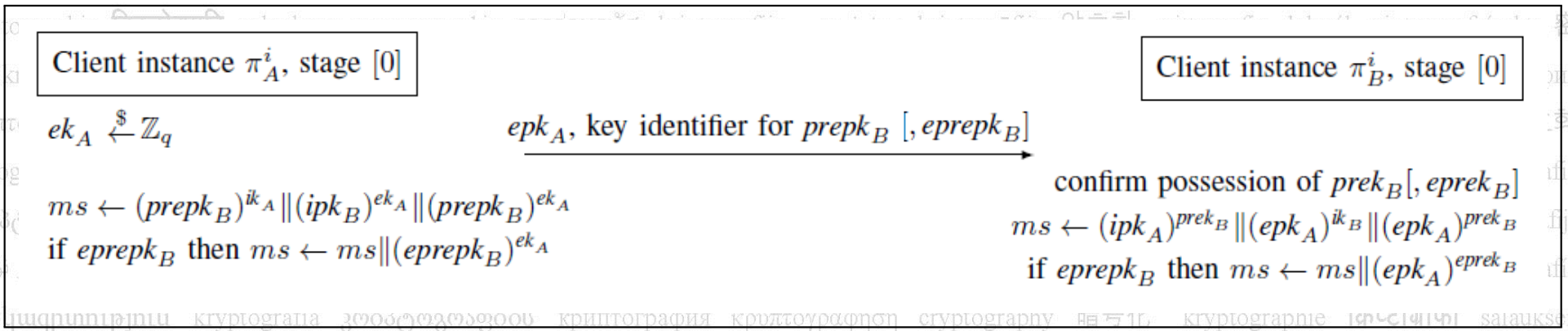
Инициализация (неинтерактивная):

- Алиса (инициатор) запрашивает данные Боба у сервера.
- Выполняется «односторонний» протокол выработки общего секретного значения ***ms*** (Master Secret).
- С помощью диверсификации из значения ***ms*** вырабатываются ключи ***rk*** (Root Key) и ***ck*** (Chain Key)



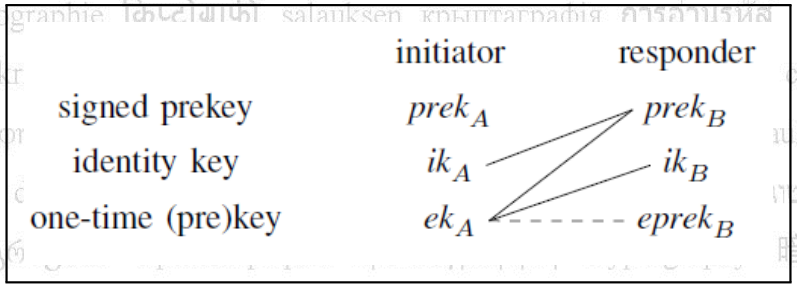
! Канал аутентифицированный !

Signal Key Exchange KDF



Алиса: $ms = (prepk_B)^{ik_A} || (ipk_B)^{ek_A} || (prepk_B)^{ek_A} [|| (eprepk)^{ek_A}]$

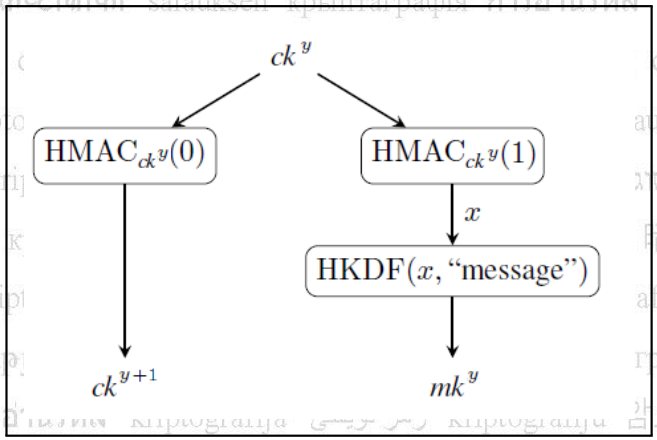
Боб: $ms = (ipk_A)^{prek_B} || (epk_A)^{ik_B} || (epk_A)^{prek_B} [|| (epk_A)^{eprek_B}]$



Assymmetric triple-DH

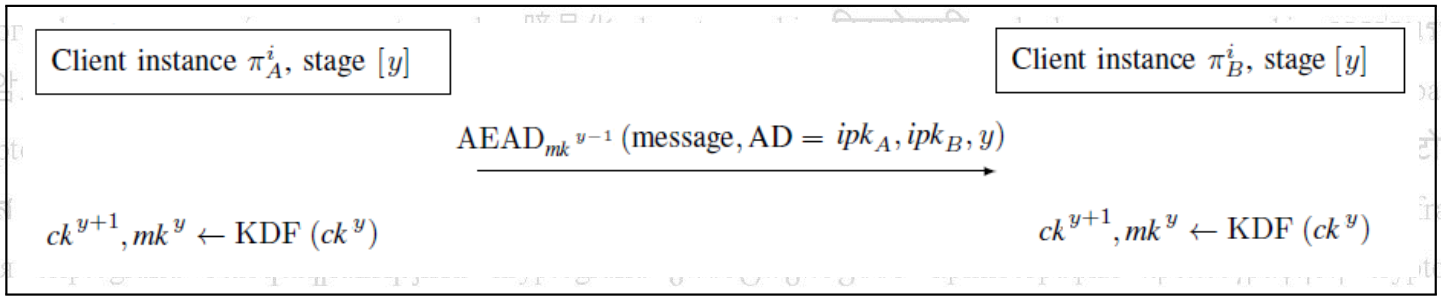
Обмен сообщениями:

- Для каждого нового сообщения из ключа ck^y (Chain Key) вырабатывается ключ mk^y (Message Key) и ключ ck^{y+1} для следующей диверсификации.

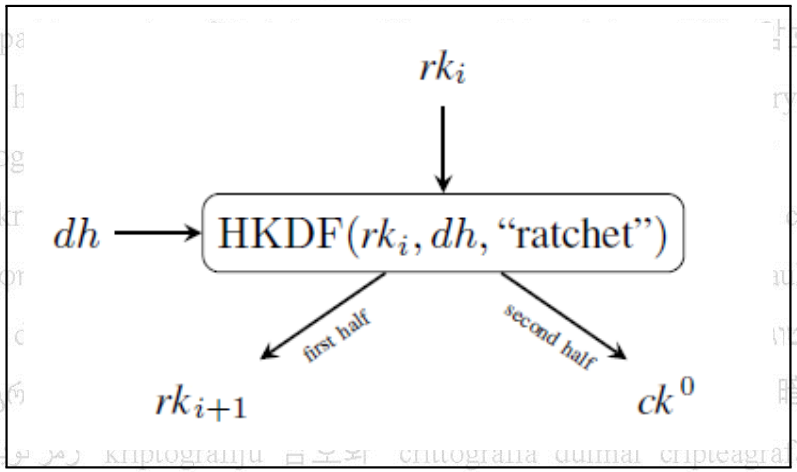


Symmetric ratcheting

- AEAD шифрование на ключе mk.



Учет нагрузки на ключ ck^0:



Assymmetric ratcheting

- Обмен новыми эфемерными значениями DH для выработки значения *dh*;
- Выработка новых значений *rk* и ck^0 с помощью функции диверсификации.

МОДЕЛЬ ПРОТИВНИКА MS-IND
(The multi-stage key indistinguishability security)

Возможности противника: активный противник, полностью контролирующий сеть.

- Инициирование нескольких параллельных сессий между любыми клиентами с любым распределением ролей;
- Перехват и подмена любых сообщений в канале;
- Частичное раскрытие секретной информации произвольных сессий $(ik, prepk, mk^y)$.

Угроза: отличение ключа mk^y (Message Key) произвольного «нераскрытого» этапа произвольной «нераскрытой» сессии от случайной строки.

Вытекающие свойства:

- Корректность;
- Конфиденциальность;
- (Неявная) аутентификация;
- Forward secrecy.

Theorem

[Cohn-Gordon et al.] В предположении вычислительной трудности задачи *GDH* (Gap Diffie-Hellman) протокол Signal стойкий в модели MS-IND со случайным оракулом (*KDF*).

$$GDH = CDH^{\mathcal{O}_{DDH}}$$

CRYPTOPRO

		WhatsApp	Signal
Open-source code		X	✓
E2E		Signal	Signal
Metadata		Вся побочная информация	Номера телефонов в хэшированном виде, время последней активности
Aut	Сообщения	key fingerprints	key fingerprints
	Звонки	key fingerprints	two words
Безопасное хранение данных		данные хранятся на устройстве в открытом виде	данные хранятся на устройстве в шифрованном с помощью пароля виде
Уязвимости		X	X

Протокол: Wickr Messaging Protocol (февраль, 2017)
Разработчик: Wickr Inc., San Francisco, US
Прототип: Wickr Professional 2015

- Регистрация пользователя
- Регистрация устройства пользователя
- Обмен сообщениями
- Хранение ключей и данных на устройстве

CRYPTOPRO

Генерация ключей и идентификаторов пользователя:

- Ключевая пара *Kr* и *PKr* (Root Identity Key Pair)
- Симметричные ключи
  - *Krs* (Remote Storage Root Key)
  - *Knsr* (Node Storage Root Key)
  - *Krbk* (Recovery Bundle Key)
- Идентификатор *IDr* (Root identifier)

User $\Rightarrow$ *PKr*, *IDr* $\Rightarrow$ Сервер

!Канал аутентифицированный!

Генерация ключей и идентификаторов устройства:

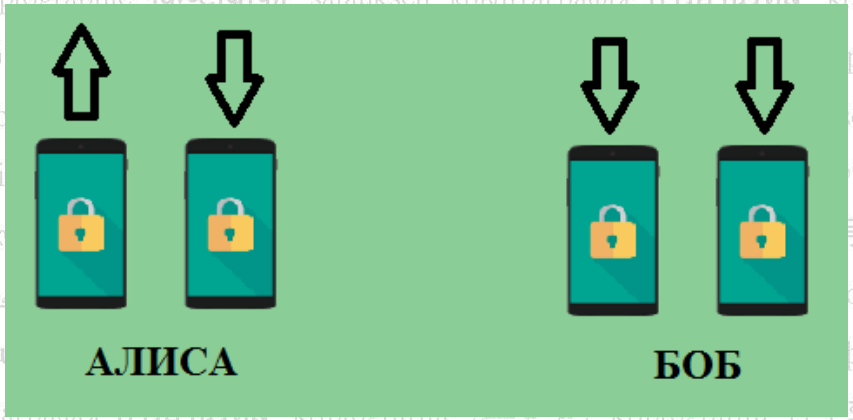
- Ключевые пары
  - Kn и PKn (Node Identity Key Pair)
  - Набор $\{KEn, PKEn\}$ (Node Ephemeral Key Pairs)
- Симметричный ключ $Klsd = KDF(Knsr||Device\_data)$ (Local Storage Device Key)
- Идентификаторы
  - IDn (Node identifier)
  - Набор $\{IDken\}$, соответствующий эфемерным ключам.

Device $\Rightarrow PKn, Sign_{Kr}(PKn), IDn \Rightarrow$ Сервер

Device $\Rightarrow \{PKEn, Sign_{Kn}(PKEn), IDken\} \Rightarrow$ Сервер

CRYPTOPRO

Подготовка ключевой информации:
Получение открытой информации от сервера (идентификаторы и подписанные открытые ключи других устройств отравителя и всех устройств получателя):



- IDr_r, PKr_r (Ключ пользователя)
- $IDn, PKn, Sign_{Kr_r}(PKn)$ (Ключ устройства)
- $IDken, PKEn, Sign_{Kn}(PKEn)$ (Одноразовый эфемерный ключ)

Передача сообщений (Node-to-node):

1. Генерация симметричного ключа $Kpayload$
2. Вычисление $Kheader = KDF(IDr_s||IDn_s)$
3. Генерация эфемерных ключей KE_s and PKE_s
4. Для каждого устройства-получателя:
   1. $Kexchn = KDF(DH(KE_s, PKEn)||PKr_s||PKr_r||IDn)$
   2. $KEDn = Enc_{Kexchn}(Kpayload)||IDn||IDken$
5. $KEL = KED1||KED2||\ldots||KEDn$ (Key Exchange List)
6. Вычисление $EPH = Enc_{Kheader}(PKE_s||KEL)$
7. Вычисление $EP = Enc_{Kpayload}(Content)$

$$\boxed{\text{Device}} \Rightarrow EP||EPH||Sign_{Kn_s}(EPH||EP) \Rightarrow \boxed{\text{Device}}$$

CRYPTOPRO

Прием сообщения $EP||EPH||Sign_{Kn_s}(EPH||EP)$:

1. Получение идентификаторов IDr_s и IDn_s и соответствующих открытых ключей PKr_s и PKn_s от сервера.
2. Проверка подписи $Sign_{Kn_s}(EPH||EP)$.
3. Вычисление $Kheader = KDF(IDr_s||IDn_s)$
4. $Dec_{Kheader}(EPH) = PKEs||KEL$
5. По IDn и $IDken$ выбор $KEn, PKEn$
6. $Kexchn = KDF(DH(KE_s, PKEn)||PKr_s||PKr_r||IDn)$
7. Вычисление $Kpayload = Dec_{Kexchn}(Enc_{Kexchn}(Kpayload))$
8. Вычисление $Dec_{Kpayload}(Enc_{Kpayload}(Content))$

CRYPTOPRO

Свойства:

- Identity hiding;
- Forward secrecy;
- Аутентификация сообщений.

Их стоимость:

- Генерация большого количества случайной информации при каждой отправке сообщения;
- Выработка общего ключа DH + 3 операции зашифрования/расшифрования + 2 операции диверсификации + 1 операция подписи/проверки подписи при каждом обмене сообщениями.

CRYPTOPRO

Хранение сообщений:

- Сообщения удаляются по таймеру.
- Сообщения хранятся в зашифрованном на ключе $Klsd = KDF(Knsr||Device\_data)$ виде.

Хранение ключей:

- Ключи $Kr, Krs, Knsr$ хранятся в зашифрованном на ключе $Krbk$ виде.
- Ключ $Krbk$ также хранится в зашифрованном виде. Для шифрования используется ключ $K = KDF(Passphrase)$.

CRYPTOPRO

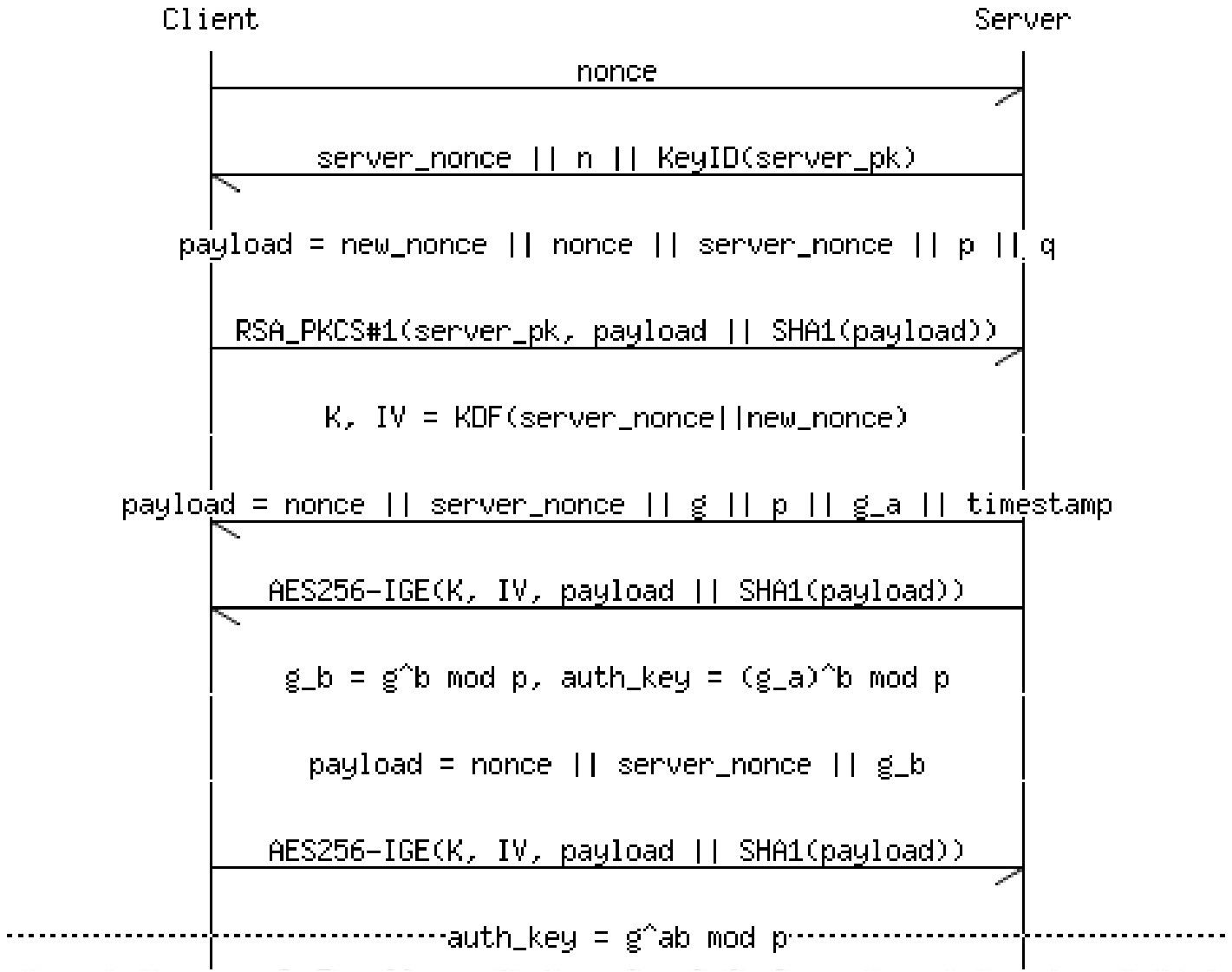
Протокол: MTProto

Разработчик: Telegram Messenger LLP, Telegram LLC.

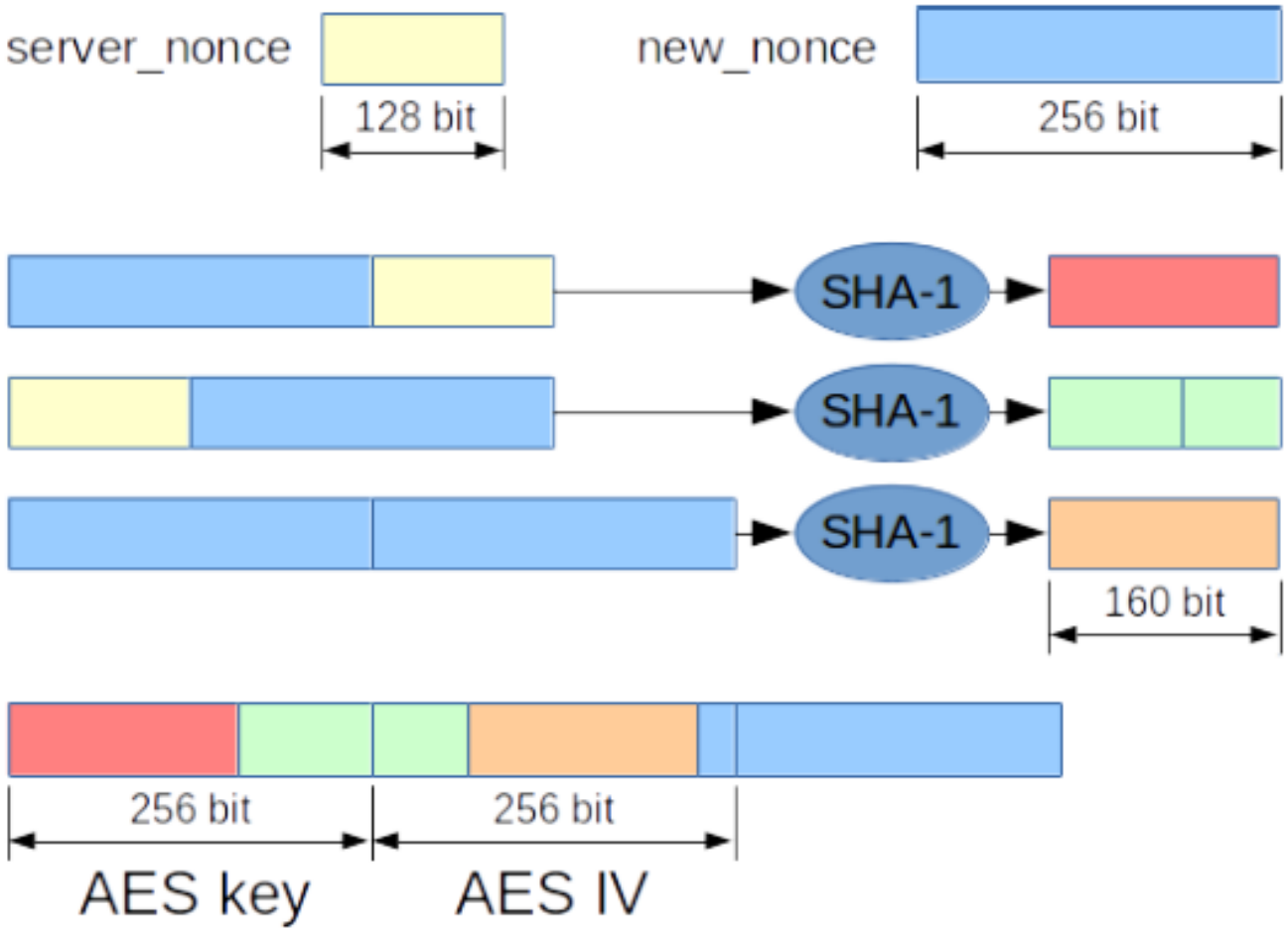
Число пользователей: более 100 млн.

- Централизованное и E2E шифрование.
- Открытый код клиента (GPLv2/GPLv3).
- Код сервера закрытый, проприетарная лицензия.

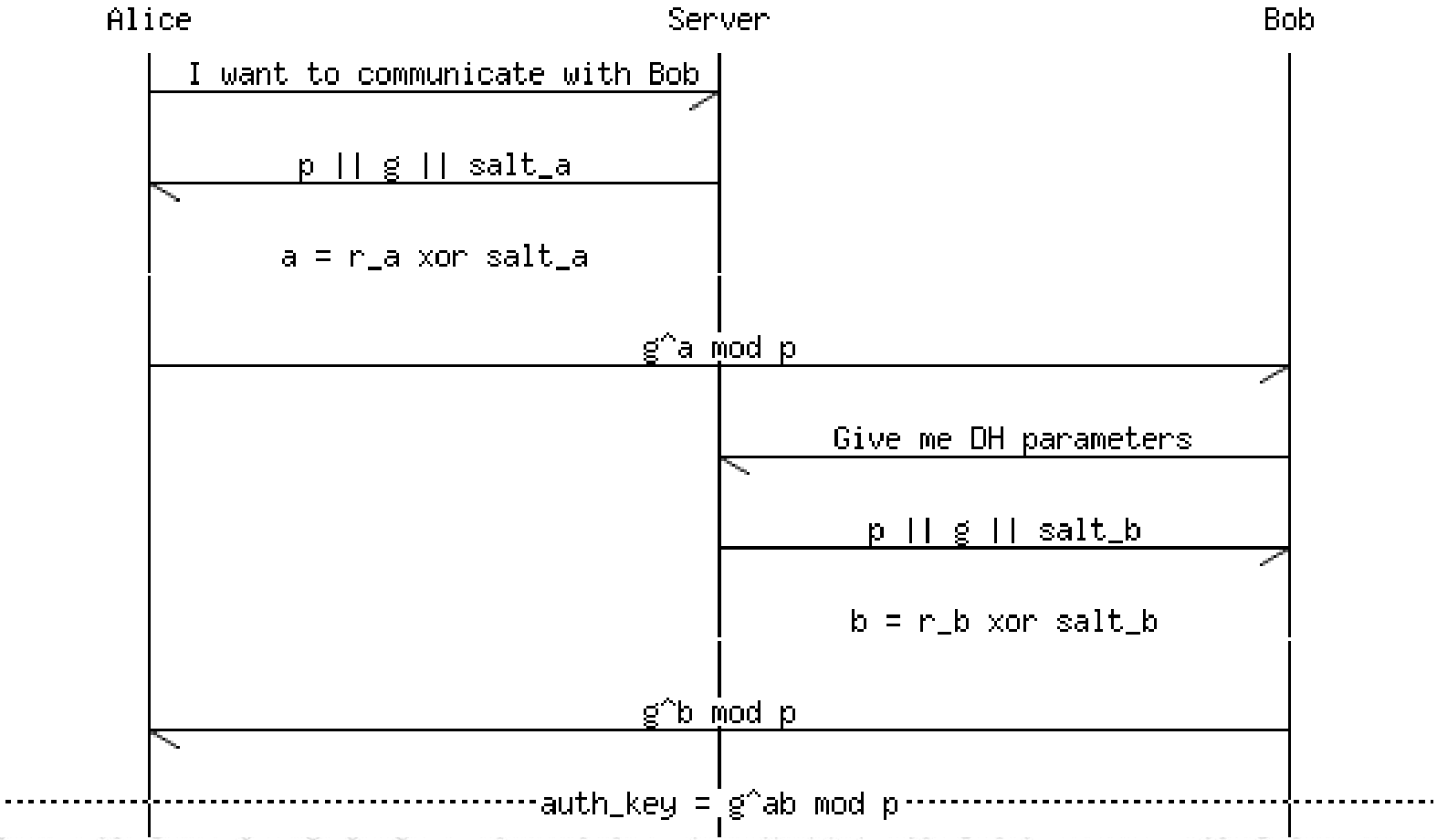
Регистрация устройства.



Регистрация устройства. Функция KDF.



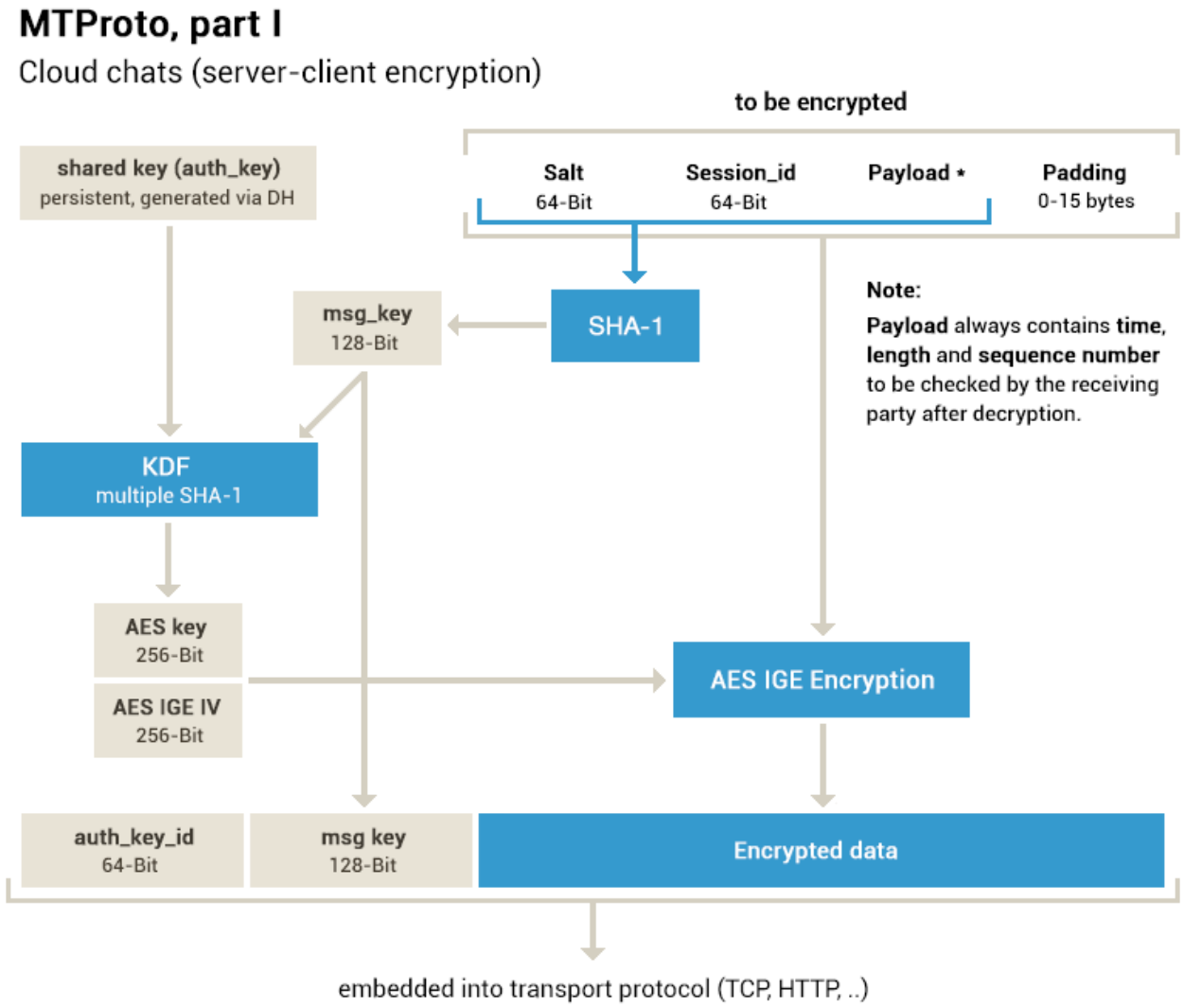
Выработка общего ключа.



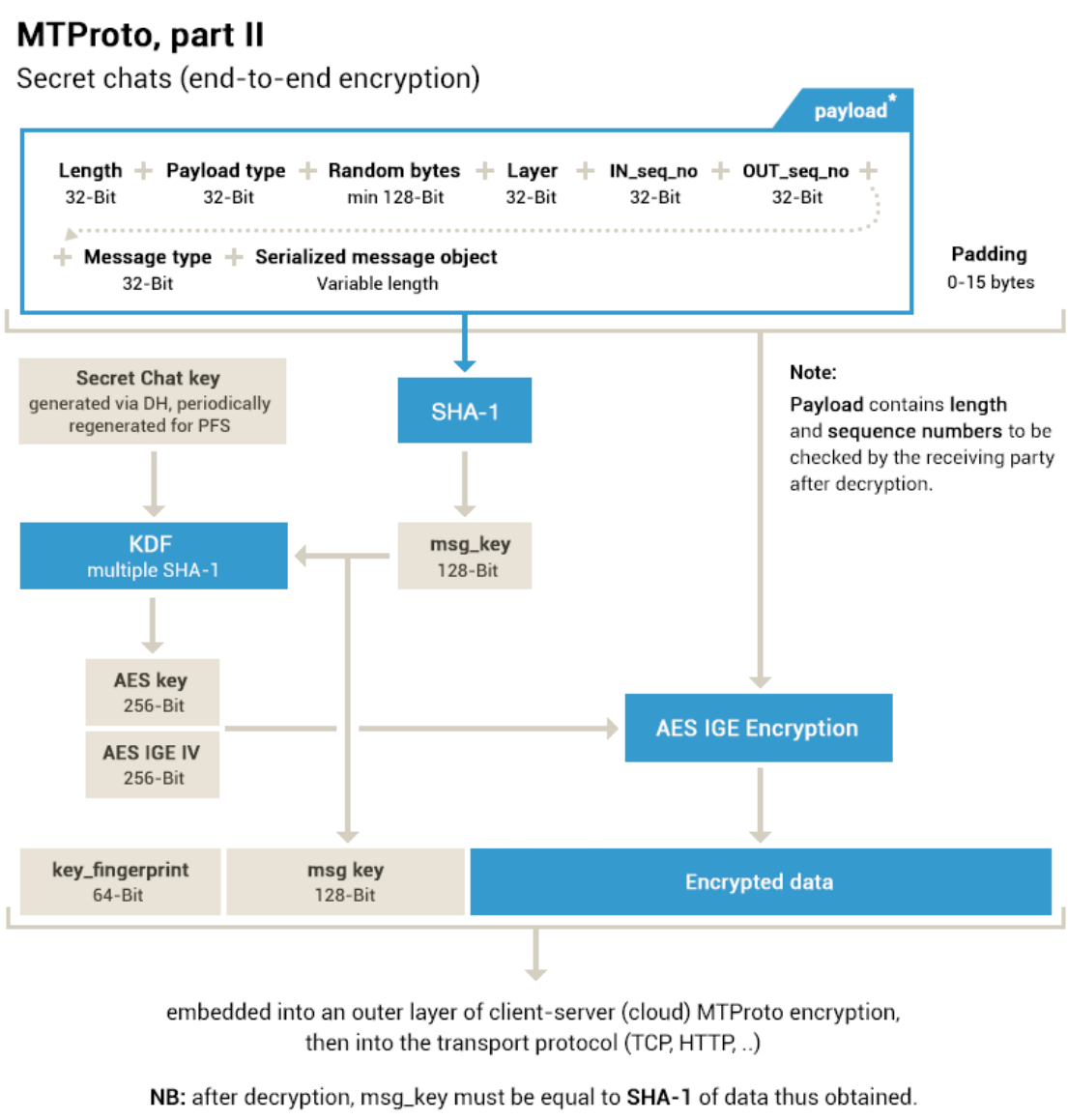
Выработка общего ключа.

$$hash = LSB128(SHA1(auth\_key))$$

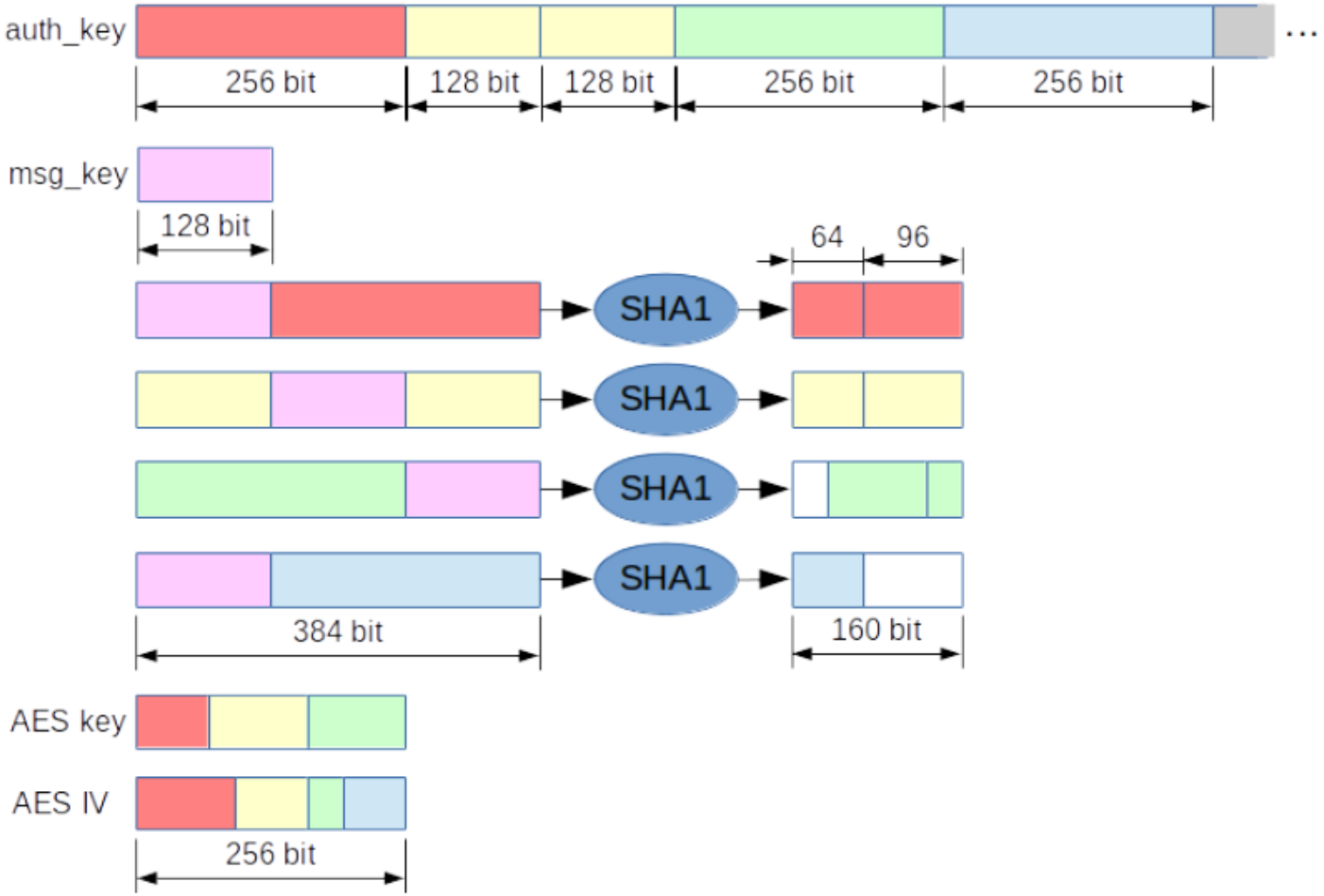
Шифрование client-server.



Шифрование E2E.



Шифрование. Функция KDF.



- Использование ненадёжных примитивов (SHA-1, AES-IGE).
- Использование неизученных конструкций (KDF как примитив, зависимость ключа от данных).
- Отсутствие анализа в адекватных моделях нарушителя.
- Игнорирование мирового криптографического опыта и его лучших практик.
- Неоднократное выявление опасно слабых конструкций в протоколе!

CRYPTOPRO

- Использование ненадёжных примитивов (SHA-1, AES-IGE).
- Использование неизученных конструкций (KDF как примитив, зависимость ключа от данных).
- Отсутствие анализа в адекватных моделях нарушителя.
- Игнорирование мирового криптографического опыта и его лучших практик.
- Неоднократное выявление опасно слабых конструкций в протоколе!

Man-in-the-middle.

Связан с использованием (практически) исходного алгоритма Диффи-Хеллмана.

- Атака практически осуществима.
- Механизм защиты неудобен (и поэтому часто используется неправильно).
- До декабря 2013 года механизм защиты легко обходился.

$$auth\_key = ((g^a)^b \bmod p) \oplus nonce$$

On the CCA (in)security of the MTProto.
J. Jakobsen, C. Orlandi, 2015.

Атака 1. Дополнение паддинга.

$$c = (tag||y_1||\ldots||y_l)$$

$$m = (aux||rnd||msg||pad)$$

$$c' = (tag||y_1||\ldots||y_l||r)$$

$$m' = (aux||rnd||msg||pad')$$

$$pad' = pad|pad^*$$

Паддинг при расчёте tag не используется!

Формальное нарушение определения, но:

1. практическая эксплуатация?
2. клиент не информирует о неуспехе расшифрования (!).

On the CCA (in)security of the MTProto.
J. Jakobsen, C. Orlandi.

Атака 2. Подмена последнего блока.

$$c = (tag||y_1||\ldots||y_l)$$

$$m = (aux||rnd||msg||pad)$$

$$c' = (tag||y_1||\ldots||y_{l-1}||y'_l)$$

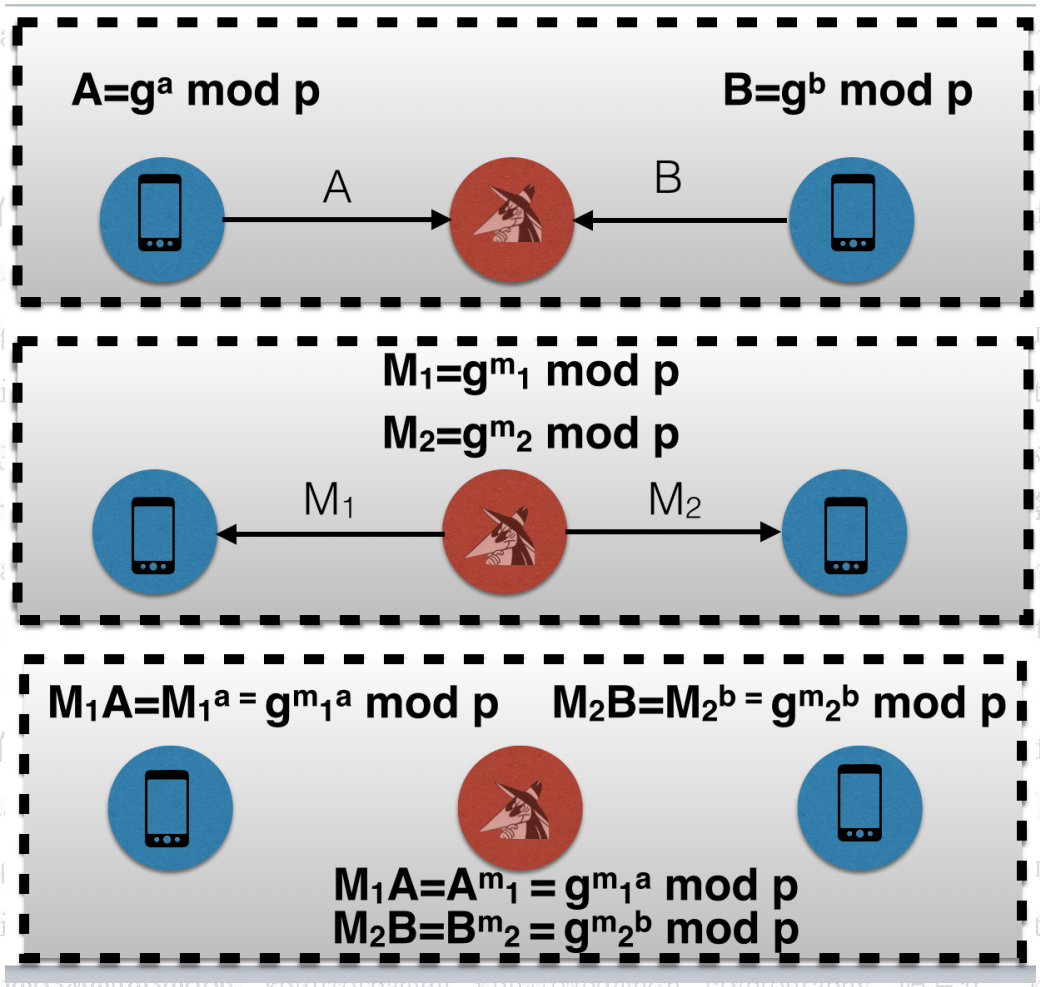
$$m' = (aux||rnd||msg'||pad')$$

Пусть длина паддинга P бит, длина блока шифра B бит.

$$P(msg = msg') = 2^{P-B}$$

$$min(P(msg = msg')) = 2^{-8}$$

Man-in-the-middle. Атака 2^{64}.



Перебираем m_1, m_2 – коллизия 128-битного отпечатка за 2^{64}!

Bard, 2006:

Шифрование в режиме IGE является CPA-стойким.

Но не BACPA-стойким!

Атака:

1. $A \to O : m_{0,1} = m_{1,1}$
2. $O \to A : c_1 = m_0 \oplus E_K(m_{b,1} \oplus c_0)$
3. $A \to O : m_{0,2} = m_{1,2}$
4. $O \to A : c_2 = m_{b,1} \oplus E_K(m_{b,2} \oplus c_1)$
5. $A \to O : m_{0,3} \neq m_{1,3}$
6. $O \to A : c_3 = m_{b,2} \oplus E_K(m_{b,3} \oplus c_2)$
7. $b = 0 \iff m_{0,1} \oplus m_{0,2} \oplus c_3 = c_2.$

Bard, 2006:

Шифрование в режиме IGE является CPA-стойким.

Но не BACPA-стойким!

Атака:

1. $A \to O : m_{0,1} = m_{1,1}$
2. $O \to A : c_1 = m_0 \oplus E_K(m_{b,1} \oplus c_0)$
3. $A \to O : m_{0,2} = m_{1,2}$
4. $O \to A : c_2 = m_{b,1} \oplus E_K(m_{b,2} \oplus c_1)$
5. $A \to O : m_{0,3} \neq m_{1,3}$
6. $O \to A : c_3 = m_{b,2} \oplus E_K(m_{b,3} \oplus c_2)$
7. $b = 0 \iff m_{0,1} \oplus m_{0,2} \oplus c_3 = c_2.$

Аутентификация сообщений.
Есть контроль целостности (хэш-значение, SHA-1, использование AES-IGE).
Аутентификации с использованием MAC – нет.

CRYPTOPRO

Своя функция KDF.

Основана на использовании хэш-функции (не HMAC!) SHA-1.

Нет анализа безопасности.

CRYPTOPRO

Perfect Forward Secrecy.

- Изначально нет.
- Начиная с версии 20 общий ключ *auth_key* пересогласовывается каждые 100 сообщений.

Спасибо за внимание! Вопросы?