Resumo do artigo de Diaz Sanches, 2019

**Seção II**

IPSEC também é um protocolo de segurança ponta a ponta notável devido à sua penetração em soluções de segurança.

IPSEC pode fornecer confidencialidade entre os nós da rede contando com a troca de chaves, ou usando chaves pré-compartilhadas. Também pode ser usado como protocolo de autenticação por meio do “Cabeçalho de Autenticação” que fornece segurança ponta a ponta equivalente ao TLS, na verdade, este é o mecanismo padrão no IPv6.

Controlar atividades mal-intencionadas de terminais, monitorar e isolar criptograficamente certos links e proteger contra falsificação de IP são várias tarefas que devem ser consideradas além da segurança de ponta a ponta.

O TLS fornece confidencialidade, autenticação e permite a negociação de quase todos os parâmetros de segurança. O TLS foi originalmente projetado para funcionar no topo de uma pilha TCP / IP e, portanto, é orientado a conexões.

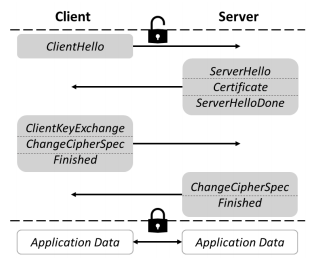
1. **Handshake in TLS**

O TLS fornece uma conexão segura sobre protocolos de transporte com autenticação opcional apenas de servidor ou mútua. Para criar o canal seguro, o TLS realiza uma troca de chaves durante o handshake para derivar uma chave secreta para proteger o canal. RSA estático é o mecanismo mais antigo e simples para a troca de chaves. Ele está disponível desde as primeiras versões do SSL. Neste mecanismo de troca de chaves, mostrado na Fig. 1, o cliente gera uma chave “pré-master”, criptografa-a com a chave pública do servidor (cujo certificado foi previamente entregue ao cliente através da mensagem “Certificado”), e então envia a chave “pré-mestre” criptografada para o servidor usando a mensagem “ClientKeyExchange”. Desta forma, o protocolo consegue trocar uma chave com o servidor de forma segura. O servidor descriptografa o “pré-mestre” usando sua chave privada. Esta prova de posse da chave privada fornece a autenticação do servidor.

A troca da mensagem “Concluída” aciona a verificação da integridade das mensagens trocadas anteriormente, que falhariam se o servidor não pudesse descriptografar a mensagem “ClientKeyExchange”.

A principal preocupação com o RSA Static é que ele não pode garantir o conceito de “sigilo de encaminhamento”, portanto, não pode garantir que as comunicações anteriores serão confidenciais no futuro. Isso acontece com qualquer mecanismo de troca de chaves em que o segredo de longo prazo usado para proteger a comunicação é uma chave compartilhada que, apesar de criptografada, é entregue através da rede. Basicamente, um invasor passivo pode armazenar a troca de chaves criptografadas junto com o tráfego criptografado que espera para quebrar ou roubar a chave privada do servidor. Se o invasor conseguir obter a chave, ele pode descriptografar todas as gravações anteriores e todas as próximas sessões protegidas com essa chave privada.

Diffie Hellman [44] (DH) fornece sigilo direto e pode ser aplicado ao TLS como mostrado na Fig. 2. Em DH, o servidor envia uma mensagem “ServerKeyExchange” após o certificado do servidor, que contém os parâmetros DH ou uma curva elíptica calculada pelo servidor. Essas informações são emparelhadas com uma chave pública efêmera gerada pelo servidor. O cliente também gera uma chave efêmera compatível com a chave do servidor e a entrega ao servidor. Dessa forma, ambos os terminais podem derivar um segredo compartilhado de longo prazo, evitando que esse segredo compartilhado seja criptografado e enviado pela rede. Além da troca de chaves, a autenticação é alcançada neste caso com a assinatura do servidor sobre os parâmetros da mensagem “ServerKeyExchange”, assim o cliente pode verificar a assinatura contra o certificado do servidor.

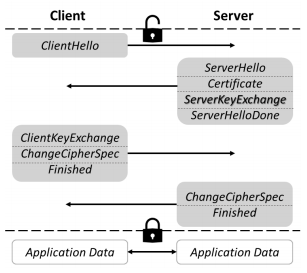


**Figura 1.** Handshake TLS com RSA Static. O cliente envia uma mensagem “ClientHello” contendo uma lista de troca de chaves suportada, conjuntos de criptografia e mecanismos de compressão para que o servidor possa forçar sua seleção com a mensagem “ServerHello”. O servidor entrega o certificado do servidor e termina a negociação usando as mensagens “Certificate” e “ServerHelloDone” respectivamente. A mensagem “ClientKeyExchange” contém a chave “pré-mestra” criptografada com a chave pública do servidor para que apenas o servidor possa descriptografar a chave “pré-mestra” e derivar a chave mestra. O cliente também entrega mensagens “ChangeCipherSpec” e “Concluído” indicando que derivou a chave mestra da chave “pré-mestra” e o tráfego futuro deve ser protegido com a chave mestra e, portanto, reconhece o servidor entregando “ChangeCipherSpec” e “ Terminado ”mensagens.

**B. Improvements in TLS Handshake Latency**

O handshake TLS requer dois atrasos de Round Trip Time (RTT) para terminar. A partir das Figuras 1 e 2, o leitor deve observar que o número de mensagens TLS a serem trocadas entre os terminais não depende da troca de chaves ou do mecanismo de autenticação. Esse tempo de handshake, junto com o handshake TCP (não se aplica ao DTLS), pode ser um tempo considerável para dispositivos que exigem uma interação rápida, como aqueles que entregam bursts de dados durante a movimentação. Isso pode acontecer com frequência em vários cenários em IoT e redes veiculares. Para melhorar a agilidade do protocolo, algumas especificações definem a abreviação de “retomada de sessão”, para que os dispositivos possam retomar sessões previamente estabelecidas com TLS. A especificação de handshake originalmente abreviada usava identificadores de sessão gerenciados pelo servidor. Outras especificações permitem que o cliente armazene “tickets de sessão” que podem ser resgatados posteriormente, evitando que o servidor armazene o estado do cliente.

No entanto, a negociação em TLS pode ter um impacto importante na eficiência do protocolo, dependendo do protocolo de transporte e dos conjuntos de criptografia selecionados. Por exemplo, dispositivos IoT / M2M, especialmente aqueles que usam interfaces de rádio restritas, devem observar que os pacotes maiores que a Unidade de Transferência Máxima (MTU) são fragmentados, aumentando a latência e o gasto de energia. Existem vários ataques que podem forçar os dispositivos a fragmentar dados em certas circunstâncias. Além disso, além da seleção do conjunto de criptografia, a negociação da compressão também pode ser crítica para evitar ataques, como será discutido na Seção II-C, mas também para melhorar a eficiência.

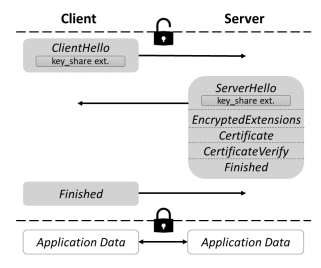


**Figura 2.** Aperto de mão TLS com curva elíptica Diffie Hellman (ECDH). Após a negociação com as mensagens hello, o servidor entrega seu certificado. A troca de chaves é então iniciada pelo servidor com a mensagem “ServerKeyExchange” que contém o material da chave ECDH, para que o cliente possa derivar uma chave efêmera. O “ServerKeyExchange” contém também uma assinatura com a chave do servidor que verifica com o certificado entregue na mensagem “Certificado”. O cliente pode derivar uma chave efêmera e entregá-la ao servidor usando ClientKeyExchange ”para que ambos os terminais possam derivar uma chave mestra de acordo com DH. Da mesma forma que o handshake apresentado na Fig. 1, nenhuma chave criptografada é entregue durante o handshake, permitindo o sigilo de encaminhamento.

O handshake abreviado mencionado anteriormente alivia o problema de retomar um canal seguro entre entidades na IoT. Por exemplo, vários aplicativos requerem dispositivos para atualizar um determinado recurso ou obter informações periodicamente. No entanto, vários outros aplicativos em IoT / M2M podem exigir a execução de solicitações de recursos que não se repetirão durante um período de tempo razoável. Portanto, a capacidade de retomar as sessões anteriores não adiciona nenhum benefício a esses aplicativos. Por esse motivo, é necessário otimizar o handshake TLS reduzindo sua latência em qualquer caso.

As versões recentes do TLS incorporam várias melhorias como a remoção do uso de RSA Static para melhorar o sigilo de encaminhamento e a redução do handshake para 1RTT ou mesmo 0RTT, dependendo do caso. O TLS tem vários subprotocolos em camadas que gerenciam a funcionalidade do TLS internamente. O subprotocolo “ChangeCipherSpec” se encarregou de sinalizar à outra parte que as próximas mensagens deveriam ser entregues criptografadas com a chave de sessão.

Este subprotocolo acionou a verificação das mensagens de handshake para verificar a integridade. Este subprotocolo pode fechar a conexão se as mensagens de handshake forem manipuladas, portanto, há uma chance de realizar um ataque de negação de serviço. Basicamente, é necessário aguardar o “ChangeCipherSpec”, vários RTTs após o início do handshake, para realizar o ataque. Devido a isso, este subprotocolo foi removido, então qualquer mensagem após “ServerHello” deve ser criptografada. Desta forma, adversários ativos, manipulando mensagens de handshake, podem ser bloqueados mais cedo.

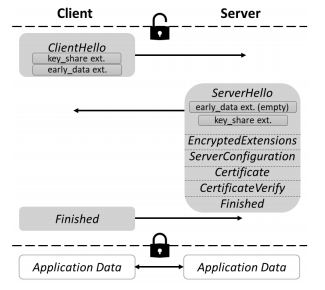


**Figura 3.** TLS 1.3 handshake 1RTT. Neste handshake TLS simplificado, as mensagens “ServerKeyExchange” e “ClientKeyExchange” foram removidas, de forma que os parâmetros DH para troca de chaves, chaves públicas e rótulos de chaves pré-compartilhadas são entregues em uma extensão chamada “key\_share”. Mais precisamente, se o estabelecimento da chave (EC) DHE estiver em uso, o cliente envia uma lista de grupos DH nomeados dentro da extensão e o servidor deve estar em um dos grupos do compartilhamento do cliente. Se assim for, após a mensagem "ServerHello", o servidor também entrega um "key\_share" contendo a chave efêmera do servidor. “EncryptedExtensions” e “CertificateRequest” contém respostas às extensões do cliente (se houver) e uma solicitação de autenticação do cliente (se a autenticação mútua estiver habilitada), respectivamente. Além disso, uma nova mensagem chamada “CertificateVerify” é usada para autenticação do servidor que contém uma assinatura sobre as mensagens de handshake trocadas até o momento.

Além disso, “ServerKeyExchange” e “ClientKeyExchange” foram substituídos pela extensão “KeyShare” para troca de chaves nas versões TLS recentes, como mostrado na Fig. 3. O resto das mensagens são mantidas na mesma ordem do protocolo original, e a autenticação do servidor é realizada assinando as mensagens de handshake anteriores. A assinatura é colocada na mensagem “CertificateVerify” para manter o TLS compatível com versões anteriores.

Versões recentes do TLS também propõem o handshake 0RTTs, apresentado na Fig. 4, que é equivalente ao handshake padrão, mostrado na Fig. 3, com a exceção de que entrega os dados do cliente em uma extensão chamada “early\_data”. Dessa forma, o protocolo de aplicativo sobre TLS pode usar essa extensão para enviar a solicitação ao servidor durante o handshake, de forma que uma vez que o canal seguro TLS seja criado, a resposta possa ser enviada. Versões mais antigas precisavam esperar até que o handshake terminasse para enviar a solicitação ao servidor, aumentando a latência. Este handshake reduzido impõe que cada mensagem após “ClientHello” seja criptografada com um segredo derivado do segredo do cliente em um “keyShareEntry” (uma entrada da extensão “key\_share”) e requer que o servidor anuncie os parâmetros semi-estáticos DH.

Deve-se observar que, apesar das solicitações de protocolo superior poderem ser entregues diretamente como parte do handshake, as propriedades de segurança do TLS são reduzidas com o handshake 0RTT.



**Figura 4.** TLS 1.3 handshake 0RTT. Se o cliente e o servidor compartilham um PSK (de um handshake anterior ou por outro meio), os clientes podem entregar dados criptografados na primeira mensagem usando a extensão “early\_data”.

Os ataques de resposta são possíveis porque o servidor deve incorporar dados aleatórios para evitar este tipo de ataques na primeira mensagem, mas neste caso, eles são entregues após a mensagem “ServerHello”. Além disso, o uso de parâmetros DH semi-estáticos pelo servidor ousa o princípio do “forward secrecy” uma vez que, pelo menos, 1RTT é necessário para estabelecer o segredo efêmero. Assim, a primeira mensagem do cliente pode não atender a esse princípio. Os parâmetros semi-estáticos DH do servidor devem ser conhecidos pelo cliente antes do handshake, devido a interações anteriores ou por qualquer outro meio, como protocolos de descoberta [50]. Em qualquer caso, é recomendável limitar sua validade a uma semana, de modo que qualquer janela de oportunidade de ataque relacionada seja reduzida.

**C. Security Considerations**

TLS e seus predecessores (SSL) sofreram dois diferentes tipo de ataques. A primeira é baseada na conceituação do protocolo e sua estrutura. O segundo, baseado na infraestrutura de chave pública, não é diretamente atribuível ao protocolo, mas o afeta, uma vez que a PKI é uma parte importante do TLS. Os problemas que a PKI traz para o TLS são explicados na Seção III e as soluções atuais também são explicadas nas seções posteriores.

O TLS não só fornece confidencialidade e autenticação opcional, mas também protege contra ataques de downgrade que pretendam impor versões anteriores (não seguras) de TLS ou o uso de um protocolo fraco. Além disso, o TLS fornece autenticação e integridade de mensagens. Esta seção descreve os ataques mais importantes relacionados à conceituação, implementação e estrutura de TLS. Alguns deles são documentados pelo IETF [51] e outros são descritos individualmente na literatura. Uma amostra dos ataques mais relevantes será apresentada agora.

O ataque de renegociação foi descoberto em 2009. Permitia realizar uma injeção de texto simples em SSL 3.0 usando o protocolo de renegociação. Basicamente, o invasor não foi capaz de descriptografar mensagens, mas sim de injetar suas próprias solicitações no início [52].

Foi resolvido incluindo uma verificação de mensagem de handshake durante a renegociação [53].

Browser Exploit Against SSL / TLS (BEAST) [54], descoberto em 2011, permitiu que um invasor contornasse a política de mesma origem (impede que um script de página entre em contato com páginas diferentes, exceto que ambas são do mesmo domínio) em TLS 1.0. Foi resolvido na seguinte versão TLS. Nesse ínterim, foi proposto o uso de RC4 como cifra de fluxo, uma vez que era imune a BEAST. Infelizmente, em 2013, uma vulnerabilidade foi descoberta alertando os implementadores contra o uso do RC4.

O RC4 não estava livre de ataques antes. Ele era imune ao BEAST, pois o TLS permitia o uso do RC4 apenas em certas circunstâncias que limitavam seu uso. As vulnerabilidades RC4 de 2013, encontraram desvios estatísticos no algoritmo que o tornaram desaconselhável para prevenir o BEAST [55]. Posteriormente, foi descoberta a possibilidade de recuperar texto simples após observar grande tráfego TLS e devido a isso, RC4 foi permanentemente proibido em TLS [56].

O vazamento de informações da taxa de compressão Made Easy (CRIME) [57], descoberto em 2012, permitiu que um invasor encontrasse mensagens de texto simples explorando o preenchimento e a compactação, sendo possível roubar cookies de autenticação. Este ataque não afetou exclusivamente o TLS, mas também afetou SPDY, HTTP e outros. Reconhecimento e Exfiltração de Navegador via Adaptive Compression of Hypertext (BREACH) [58], que foi baseado em CRIME, foi apresentado em 2013. O BREACH permitiu que invasores extraíssem informações confidenciais, incluindo informações de autenticação, após observar certos dados. Isso permitiu que os invasores redirecionassem a vítima para sites maliciosos ou mesmo injetassem conteúdo nas páginas da Web acessadas por meio do canal criptografado. Considerando que a proteção contra CRIME foi possível eliminando a compressão TLS e cabeçalhos SPDY, as implementações atuais de TLS ainda são vulneráveis a BREACH, uma vez que não é viável eliminar a compressão de protocolos de aplicativos.

Em 2013, foi apresentado um ataque que permitiu aos atacantes bloquear mensagens de logout, injetando uma mensagem de terminação TCP (TCP FIN) sem o conhecimento da vítima [59]. Para alcançar o resultado, não era necessário infectar a máquina do usuário, mas comprometer um hotspot ou qualquer outro elemento de rede no caminho.

As primeiras versões do SSL eram vulneráveis ao “ataque de padding oracle” descoberto em 2002, que permitia usar um servidor (oracle) para saber se o padding estava correto ou não, permitindo decifrar mensagens com a chave do servidor sem seu conhecimento. O ataque foi viável usando CBC. Em 2013, uma variante do ataque de preenchimento, denominado Lucky Thirteen [60], permitia quebrar a autenticação da mensagem em TLS analisando o tempo gasto na criptografia (timing side-channel attack). Ele foi resolvido com uma extensão da especificação TLS [61]. O Padding Oracle On Downgraded Legacy Encryption (Poodle) foi apresentado em 2014, mostrando como o Cipher Block Chaining (CBC) em SSL3.0 é vulnerável a ataques de preenchimento. Apesar de a maioria dos servidores usar, pelo menos, TLS 1.0, o ataque exigiu forçar o uso de SSL3.0 como fall-back, frequentemente suportado até 2015 [62].

Em 2014 e 2015, duas vulnerabilidades críticas, conhecidas como HeartBleed e BERSerk, que afetaram o OpenSSL e outras implementações amplamente adotadas, foram apresentadas. O Heartbleed permitiu que os invasores explorassem um bug para extrair dados dos servidores; O BERSerk explorou um bug no ASN.1 que permitia ataques man-in-the-middle em várias implementações.

Factoring RSA Export Keys (FREAK), identificado em 2015, explora uma antiga restrição à exportação SSL / TLS introduzida pelo governo dos Estados Unidos. Essa restrição limitava o tamanho das chaves RSA a 512 bits. Em 2010, foi demonstrado que quebrar chaves RSA curtas era simples o suficiente para se tornar um problema. FREAK contou com um ataque de rebaixamento. Ele forçou a vítima a usar uma versão antiga abandonada de SSL com suporte à restrição. O ataque consistia em influenciar a negociação do conjunto de cifras para obrigar o uso de algoritmos fracos [63]. Logjam, também de 2015, era semelhante a FREAK, mas reforçando uma velha restrição relacionada a Diffie Hellman.

Descriptografar RSA com Obsolete and Weakened eNcryption (DROWN) [64], anunciado em 2016, poderia ser usado para atacar servidores usando uma combinação de versões em vez de uma versão concreta. Ele usou um ataque de texto cifrado escolhido adaptável combinado com um downgrade para SSLv2 (que ainda era suportado por muitos servidores). O ataque foi útil para reduzir o esforço necessário para um ataque man-in-the-middle. Pode-se estimar que 33% dos servidores em 2016 foram afetados.

III. TLS/PKI PROBLEMS AND SECURITY EVOLUTION

**Seção 4**

Como já foi discutido, a PKI não possui uma autoridade raiz única capaz de verificar os certificados, caso contrário, ela possui um grande conjunto de autoridades raiz independentes capazes de verificar apenas certificados irmãos. A lista de CAs raiz é compilada por fabricantes de software ou sistemas operacionais.

A Seção III-A explica o crescimento desproporcional de ACs e seus problemas associados. Nesta seção, o conceito de Certificate Pinning é apresentado como uma ferramenta para evitar ou diminuir o principal problema da Web atual que, sem dúvida, afetará também os serviços de repouso IoT e M2M: a falta de confiança.

Para ilustrar melhor o problema, considere a autenticação TLS. No processo de autenticação de um servidor, o cliente inicia um handshake TLS e obtém o certificado do servidor durante esse handshake, conforme declarado na Seção II. Existem duas operações distintas durante um handshake: derivar uma chave para proteger o tráfego de maneira segura e autenticar o servidor ou o cliente e o servidor (mútuo). Apesar de existirem vários algoritmos de autenticação, que podem ser negociados durante o handshake, a parte a ser autenticada deve fornecer um comprovante de posse da chave privada associada ao certificado. O processo é equivalente para autenticação de servidor e mútuo, por isso vamos nos concentrar na autenticação de servidor.

O cliente, uma vez de posse do certificado do servidor, verifica a assinatura e constrói uma cadeia de certificados PKI do certificado do servidor até a primeira CA intermediária ou raiz em que ele confia, verificando a lista de CA confiáveis em uso. Se essa verificação for bem-sucedida, o cliente também pode verificar a lista de revogação de certificados publicada pela CA. Se a verificação for bem-sucedida e o certificado não for revogado, ele verifica se o nome alternativo do certificado corresponde ao nome de domínio do servidor. Se for igual, a conexão é considerada confiável.

Portanto, a criação da cadeia de certificados PKI é a parte mais fraca da verificação. É sabido que CAs públicas e privadas, que estão na lista confiável e, portanto, são confiáveis para os clientes, introduziram CAs irmãs intermediárias que são, portanto, confiáveis pelos clientes. A finalidade dessas CAs intermediárias pode variar de CAs subordinadas emprestadas a empresas, podem emitir certificados sem a intervenção da CA pai, a aceleradores SSL / TLS que podem acessar o tráfego em texto não criptografado. No primeiro caso, as CAs intermediárias aumentam o comprimento da cadeia de certificados, mas são transparentes para o usuário. No entanto, conforme mencionado na Seção III-A, casos como o CA “C = DE, CN = Deutsche Telekom Root CA 2” que tinha 252 sub CAs em 2011 e “C = US, CN = GTE CyberTrust Global Root” tinha 93 sub CAs, levantem preocupações sobre o controle que o CA pai tem sobre os CAs irmãos.

No segundo caso, vários operadores de rede introduzem CAs intermediários que permitem às empresas acelerar o tráfego SSL / TLS [84]. Por meio dessas CAs intermediárias, os provedores podem emitir certificados de servidor intermediário que interceptam e aceleram o tráfego SSL / TLS em nome dos clientes (por exemplo, servidores da Web com grande tráfego). Dessa forma, os servidores podem transferir o gerenciamento de fluxos criptografados para terceiros. No entanto, apesar de um cliente que acessa um servidor perceber uma melhora no tempo de resposta, ele geralmente não tem conhecimento do fato de seu tráfego não ser protegido de ponta a ponta, mas descriptografado em um ponto intermediário da rede e entregue em texto não criptografado de que apontam para o servidor. Além disso, existem outros casos preocupantes em que essas soluções são incorporadas a dispositivos eletrônicos de consumo que podem violar a privacidade do usuário

Qualquer uma dessas numerosas CAs raiz ou intermediárias, que podem estar comprometidas ou mal utilizadas, podem emitir certificados para qualquer nome de domínio sem o conhecimento do proprietário do domínio. Existem vários casos bem conhecidos que ocorreram sem serem conhecidos até mais tarde. Em 2011, o Instituto de Pesquisa e Desenvolvimento Agrícola da Malásia CA foi comprometido e usado para criar uma ferramenta maliciosa a partir da ferramenta de atualização do Acrobat. Essa ferramenta instalou atualizações que pareciam legítimas, mas que transformaram o cliente em uma máquina espiã sob o controle de hackers. Até que o problema fosse detectado e o certificado CA fosse revogado, eles poderiam ter personificado qualquer domínio da Internet.

Em 2011, uma CA chamada Diginotar foi usada para emitir certificados para Gmail e Facebook, entre outros. O Google anunciou o problema por meio de um comentário de um cliente dos Grupos do Google. Outros casos, como Comodo em 2011 e TurkTrust em 2013, também foram exemplos muito populares de CAs comprometidos. Recentemente, a TrustWave admitiu ter emitido certificados CA raiz subordinados para clientes que eram capazes de emitir certificados PKI para qualquer domínio do planeta sem o controle da TrustWave. Essas práticas aumentam o risco de encontrar certificados emitidos por terceiros sem o conhecimento dos proprietários dos domínios que são falsificados. Esses ataques não quebram, mas modificam a cadeia de confiança e funcionam de forma transparente para o usuário e o proprietário do domínio, portanto, são bastante perigosos

Apesar de vários protocolos terem sido propostos para gerenciar a confiança, ou melhor, para gerenciar "âncoras de confiança" [87], [88] para a construção da cadeia de certificados, os problemas decorrentes do comprometimento de uma CA ainda estão presentes, uma vez que o certificado malicioso é emitido por uma CA que está direta ou indiretamente na lista de CA confiáveis e, como foi argumentado, há muitas.

“Pinning de certificado” é um conceito que permite aos clientes obter uma melhor certeza de que um certificado usado por um servidor não está comprometido. Nas seções a seguir, várias propostas de “Pinning de certificado” serão avaliadas. A Transparência dos Certificados, descrita na Secção IV-A e SK, na Secção IV-B, propõe infraestruturas complementares para o controlo global dos certificados em conjunto com uma verificação cruzada do cliente; Trust Assertion for Certificate Keys (TACK), analisado na Seção IV-C, propõe uma verificação cruzada controlada pelos proprietários do domínio; A autorização da autoridade de certificação do DNS (CAA), descrita na Seção IV-D, permite que o proprietário do domínio limite quais CAs podem emitir certificados para seu domínio; HTTP Strict Transport Security e HTTP Public Key Pinning Protocol descrevem novos cabeçalhos HTTP que impõem políticas para TLS e transportam a cadeia de certificados a ser usada (Seção IV-E); finalmente, DANE junto com DNSSEC são descritos na Seção IV-F2. As soluções serão comparadas na Seção V e sua viabilidade para cenários IoT / M2M será discutida.

1. Transparência de certificado

O Certificado de Transparência (CT) [89] foi proposto como uma contramedida à falsificação de identidade de sites. O CT fornece um “Certificado de Pinning” ou verificação alternativa para usuários e um sistema de vigilância para CAs. Basicamente, permite que uma estrutura verificável contendo rastros de certificados de servidor existentes seja auditada por vários atores, cada um deles com seus próprios interesses, para detectar CAs maliciosos ou comprometidos

Os objetivos desta proposta são: proteger a emissão de certificados de CAs maliciosos para um determinado domínio sem o conhecimento do proprietário do domínio; fornecer um sistema de auditoria e monitoramento para permitir que os proprietários de domínio detectem as emissões não autorizadas de certificados; e, como consequência do anterior, protege os usuários de serem enganados.

A estrutura verificável consiste em uma Árvore Merkle (MT) [90]. Essa árvore contém um hash de um objeto sujeito a verificação em cada folha. A existência de um objeto na árvore e a ordem em que foi adicionado à árvore podem ser verificadas por meio do TM. Para fazer isso, os nós pais em um MT contêm um hash que combina os hashes de seus filhos e continua até a raiz, que contém um hash combinando o hash de cada descendente. Desta forma, qualquer mudança no conteúdo ou na ordem das folhas, altera o valor da raiz.

Verificar uma folha em um MT requer o processamento de um número de nós proporcional ao logaritmo do número de nós [91], portanto, verificar um único objeto dentro de uma árvore de um milhão de folhas requer o processamento de 20 nós [92]. A árvore utilizada na TC é baseada no método proposto por Crosby e Wallach [93], que utiliza um hash SHA-256. Cada nó é calculado sobre uma lista de dados e o hash tem 32 bytes de comprimento. Assim, para uma sequência ordenada de n entradas D [n] = {d (0), d (1), ..., d (n - 1)} o Hash da árvore Merkle (MTH) (MTH ()) é definido da seguinte forma [89] para uma sequência vazia, um único elemento e n elementos

Uma sequência vazia MTH () = SHA-256 (). Para uma única entrada na lista correspondente a uma folha de árvore, MTH (d (0)) = SHA256 (0x00 || d (0)). A razão para concatenar 0x00 ed (0) permite diferenciar a operação de hash sobre as folhas do resto dos nós, que são concatenados com 0x01. Caso contrário, seria possível ou mais fácil gerar colisões ou segundas pré-imagens do hash [94].

Se n> 1, considere k a maior potência de dois menor que n, então k <n≤2k. O MTH de uma lista de n elementos, D [n], definido de forma recursiva é MTH (D [n]) = SHA-256 (0x01 || MTH (D [0: k]) || MTH (D [ k: n])) onde || significa concatenação e D [k1: k2] = d (k1), d (k1 + 1), ..., d (k2 - 1) é uma lista de elementos k2 - k1. Desta forma, uma assinatura sobre o MTH resultante pode ser usada para verificar toda a árvore.

No CT, cada folha armazena um certificado emitido por uma CA mediante solicitação de CAs. Existem duas verificações interessantes na árvore. O primeiro, verifica se determinado certificado pertence à árvore, ou seja, foi comunicado pela CA correspondente e adicionado. A segunda busca verificar se essa árvore somente de acréscimos não foi adulterada, de modo que a ordem em que os certificados foram adicionados à árvore seja consistente com seus carimbos de data / hora. Para provar isso, o CT define “Merkle Audit Path” e “Merkle Consistency Proof”.

O Merkle Audit Path (MAP) para uma determinada folha é definido como a lista mais curta de nós da folha à raiz que permite derivar o MTH para aquela árvore. Um verificador de posse do MTH assinado, usa o MAP para verificar se uma folha pertence a uma árvore. Assim, se o MTH derivado do MAP corresponder ao MTH em posse do verificador, o MAP prova que a folha pertence à árvore

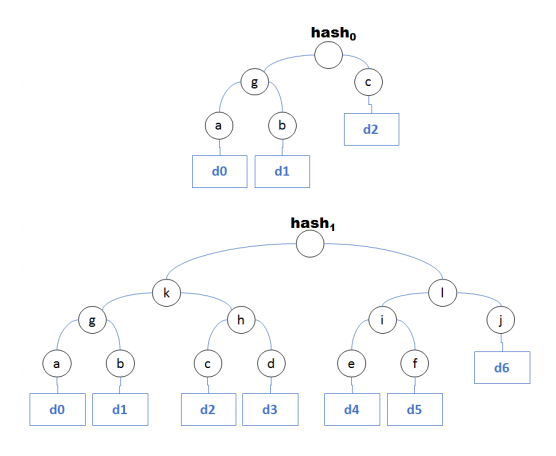


Fig. 8. Exemplo de árvore Merkle em dois instantes de tempo diferentes para ilustrar os exemplos de cálculo de caminho de auditoria Merkle (MAP) e prova de consistência Merkle (MCP).

O Merkle Consistency Proof (MCP) verifica se a árvore é somente anexada para que as folhas existentes não sejam modificadas ou excluídas e mantidas ordenadas. Suponha que um verificador tenha o hash raiz atual, MTH (D [n]), e um hash raiz anterior correspondente às primeiras m folhas, MTH (D [0: m]), com m ≤ n. MCP é a lista mínima de nós da árvore que permite verificar se as m primeiras folhas são iguais em ambas as árvores.

A Fig. 8 mostra a mesma árvore em dois momentos diferentes. O primeiro tem um hash rotulado como hash0 e o segundo hash1. Para a árvore com hash1, o MAP para d0 será a lista [b, h, l], pois d0 pode derivar a, mas precisa de b para calcular g, h para calcular k e finalmente l para calcular hash1. d0 pertence à árvore se o hash1 corresponder ao MTH esperado. Equivalentemente, MAP para d4 será [f, j, k].

Se ambas as árvores forem consideradas, uma prova de consistência entre a árvore com hash0 e hash1 será MCP = [c, d, g, l]. c e g serão usados para verificar o hash0, d para verificar k e l para verificar o hash1. Desta forma, pode-se verificar que hash1 foi gerado a partir de hash0 e, portanto, é consistente.

CT define três componentes que são servidor de log, monitor e auditor. Os servidores de log protegem CT MTs. Apesar de não ser especificado o número de servidores de log necessários para lidar com a Internet atual (nem quem é o responsável por eles) [89], algumas pesquisas indicam que são necessários cerca de mil servidores em todo o mundo, [95] que podem ser gerenciados por CAs, provedores de serviços de Internet e outras partes.

Quando um certificado válido é enviado a um servidor de log, ele emite um “Selo de certificado assinado” (SCT). O SCT é uma promessa do servidor de log de incorporar o certificado à árvore em um tempo menor que o “Atraso Máximo de Mesclagem” (MMD). O certificado será incluído em uma folha da árvore em uma estrutura que inclui o certificado e o SCT. Em seguida, um hash é calculado sobre essa estrutura e adicionado à árvore.

Cada vez que um certificado é adicionado à árvore, o hash de cada nó afetado pela alteração é recalculado até a raiz. O hash resultante na raiz é então assinado levando ao “Cabeçalho de árvore assinado” (STH). Portanto, apenas STH (um por árvore) e SCTs (um por certificado) são assinados com o par de chaves do servidor de log para permitir a verificação segura da árvore.

Para serem aceitos por um servidor de log, os certificados candidatos devem passar por uma verificação de PKI. Ele requer a construção de um caminho de certificação do certificado para uma CA confiável, de acordo com a lista de CAs confiáveis aceitos pelo servidor de log. Da mesma forma, todo servidor de log deve publicar sua lista de CA confiáveis [89]. O procedimento para CAs é diferente e requer pré-certificados com extensões envenenadas, conforme explicado posteriormente nesta seção.

Desta forma, não há espaço para certificados autoassinados ou aqueles emitidos por CAs locais (ilhas de segurança), bem como outros casos de uso que modificam localmente a âncora de confiança para lidar com uma alta dinâmica, como aqueles descritos posteriormente na Seção IV -F2.

Monitores são entidades que inspecionam e verificam a operação de um servidor de log. Os monitores têm interesses particulares, procurando certificados para um determinado domínio ou conjunto de domínios, mas podem ignorar outros. Cada monitor deve inspecionar cada nova entrada em cada servidor de log monitorado e pode manter uma cópia de todo o log. Portanto, os monitores devem obter periodicamente as entradas do servidor de log e o STH, verificar a assinatura e realizar verificações de consistência.

Finalmente, os auditores obtêm informações parciais de um servidor de log como entrada e verificam se as informações são consistentes com as evidências coletadas anteriormente. De acordo com a especificação, um auditor pode ser um cliente TLS ou uma entidade independente que presta serviços a clientes TLS. Basicamente, um auditor verifica a consistência de dois SCTs da mesma árvore, no mesmo servidor de log, com uma verificação de consistência (MCP).

Em geral, os auditores têm como objetivo fazer parte dos clientes TLS [95], portanto, a verificação das SCTs é delegada a eles. Isso é importante porque os clientes TLS devem rejeitar certificados sem um SCT válido. No entanto, sugere-se que os monitores possam não apenas verificar a integridade do log e procurar um interesse, mas também fornecer serviços gratuitos ou pagos para CAs e proprietários de domínio. Além disso, os monitores podem ser operados por proprietários de domínio e até mesmo atuar como auditores em nome dos clientes TLS. A operação do TC é apresentada na Fig. 9.

O CT requer que monitores com interesses particulares avisem os proprietários dos nomes de domínio monitorados para que os proprietários possam tomar as ações apropriadas quando um log for mal utilizado. Deixar de inserir um certificado no tempo de MMD ou violar a ordem na árvore (consistência) é considerado uso indevido. Os auditores podem detectar inserções com falha solicitando MAPs para cada SCT observado. A cooperação dos auditores para detectar violações de consistência pode ser alcançada por um protocolo de “fofoca”, conforme sugerido na especificação.

O suporte ao CT implica que os servidores entreguem o SCT junto com o certificado, para que os clientes possam fazer a verificação cruzada dos SCTs em relação aos certificados. Como os servidores de log são monitorados por monitores e invalidados por uso indevido, verificar os SCTs pode ser suficiente para proteger a representação usando certificados de uma CA comprometida. CT propõe várias alternativas para entregar SCTs com certificados [89], [95].

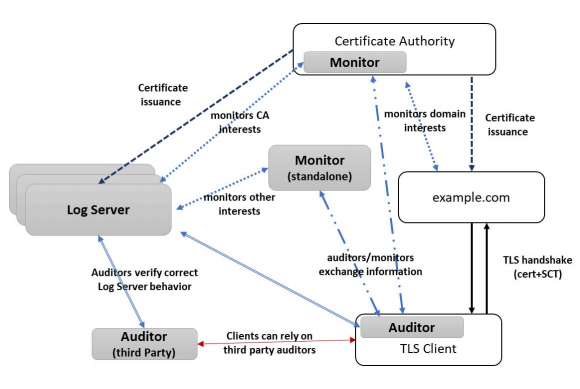


Fig. 9. Interação entre os participantes do Certificate Transparency. A emissão do certificado é uma operação única. O handshake TLS é síncrono, enquanto o restante das interações (auditor-monitor, auditor-log, monitor-log) são assíncronas.

• Incorporado no certificado: o SCT pode ser incorporado no certificado usando uma extensão X.509v3 [96]. O problema é que o SCT é obtido depois que o certificado é aceito pelo servidor de log, e incorporar o SCT ao certificado posteriormente invalidará a assinatura. Por esse motivo, as ACs que desejam incorporar o SCT no certificado, devem enviar um pré-certificado idêntico ao certificado que será eventualmente emitido. Uma extensão envenenada será adicionada ao certificado usando uma extensão X.509 crítica e assinada (TBSCertificate [24]). Dessa forma, o pré-certificado serve como uma entrada de registro, portanto, pode ser usado para verificar, mas não pode ser entregue em um handshake TLS. Este mecanismo não requer mudanças nos clientes ou servidores.

• Extensão TLS: neste caso, a estampa de tempo (SCT) é entregue durante o handshake TLS separadamente do certificado por meio de uma extensão TLS [19] com um tipo específico. Este mecanismo requer que extensões TLS sejam suportadas pelos terminais, portanto, requer uma mudança em ambas as entidades, apesar de hoje em dia as extensões TLS serem bem suportadas.

• Grampeamento OCSP: o carimbo de data / hora é entregue usando a conhecida extensão TLS “Certificate Status Request” [97]. Para indicar que essa extensão realmente contém um SCT em vez de informações OSCP, um Identificador de Objeto especial é usado.

Considerações de segurança para o cliente: A especificação não indica como os servidores de log publicam ou distribuem as chaves necessárias para verificar as assinaturas SCT. Na melhor das hipóteses, essa lista de chaves públicas do servidor de log deve ser distribuída com o software ou obtida usando outro protocolo fora de banda. Portanto, ele requer uma lista de confiança adicional além da PKI.

A detecção de falha na adição de um novo certificado requer que os clientes solicitem um MAP por SCT observado. Isso levanta questões de privacidade, pois as partes envolvidas podem rastrear o cliente observando os SCTs solicitados. Para superar o problema, os clientes podem usar auditores terceirizados confiáveis, por exemplo, do ISP. Além disso, os SCTs podem ser verificados em lotes de forma assíncrona, de forma que o horário em que o SCT foi obtido não possa ser informado por outras partes. No entanto, isso não impede que as partes aprendam as SCTs, a menos que sejam alteradas com ruído, o que requer um esforço extra do lado do cliente.

Qualquer certificado fornecido e seu SCT associado podem ser verificados usando um Cabeçalho de Árvore Assinado, do mesmo servidor de log, que foi assinado Atraso Máximo de Mesclagem (MMD) após o registro de data e hora SCT. Para verificar o SCT, basta solicitar um MAP ao log. No entanto, há uma janela de oportunidade para um ataque que depende do MMD, ou seja, o tempo que um servidor de log espera para acumular solicitações de inserção, para que possam ser adicionadas em lote, em vez de economizar individualmente em operações criptográficas dispendiosas. Devido a isso, o CT é um sistema capaz de detectar problemas com o certificado em horas [95], mas sua eficácia em ambientes dinâmicos é inversamente proporcional ao MMD. Além disso, a fim de detectar servidores de log com comportamento inadequado, o sistema exige uma adoção global

Além disso, não há uma indicação clara do comportamento esperado das CAs em relação aos servidores de log a serem usados. Por um lado, afirma-se que as CAs podem usar alguns logs quando for conveniente [95]; por outro lado, diz-se que as CAs irão solicitar a adição de um certificado a cada servidor de log disponível [92]. Em qualquer caso, é acordado que não haverá sincronização entre os diferentes servidores de log, levando a uma árvore diferente por servidor de log. Isso complica os monitores para encontrar um log capaz de verificar um determinado certificado.

Quanto ao esforço do cliente, ele deve ter acesso a uma lista de chaves públicas do servidor de log para permitir ao auditor verificar a assinatura do SCT, ou ter uma forte relação de confiança com um auditor externo, o que na prática é o mesmo problema. Além disso, um certificado verificado com Transparência de certificado também deve passar por uma cadeia de certificados PKI e validação de status de certificado [98].

B. Chaves Soberanas

Assim como Transparência de certificado, “Criptografia de chave soberana para domínios da Internet” propõe uma estrutura pública verificável e auditável somente append [92] que associa cada certificado a uma chave soberana (SK). Um SK pode ser associado a um ou mais certificados e usado para assinar o certificado final.

Um dos objetivos do SK é proteger os clientes contra ataques Man In The Middle (MITM) ou falsificação de identidade. Para atingir esse objetivo, os clientes que oferecem suporte ao SK devem verificar se a chave pública pertencente a um certificado, usada por um servidor, foi assinada de forma cruzada com o SK registrado para o domínio do servidor.

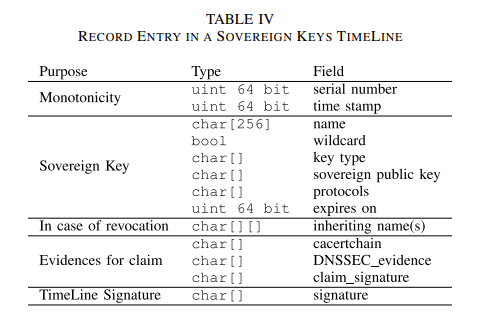
Ao contrário do CT, o SK possibilita a definição de uma rota alternativa para o servidor em caso de personificação, ataque MITM ou bloqueio de conexão (todos eles detectados pelo SK). Desta forma, além de alertar o usuário, que pode não ser eficaz [77], [99] já que os alertas são frequentemente ignorados se não houver uma forma alternativa, o SK disponibiliza outras rotas para chegar ao serviço.

A estrutura verificável usada pelo SK consiste em um apêndice “TimeLine” cujas entradas são relações entre nomes de domínio e SKs (chaves), sendo SKs diferentes daqueles usados por certificados de servidor.

O objetivo do TimeLine é armazenar e preservar o histórico das relações do domínio SK. A última entrada para um domínio, junto com o conjunto de atualizações, mudanças nos domínios de serviço, renovações, revogações e outros ajustes, é válida. Uma solicitação de um novo SK, que ficará armazenado na estrutura, contém os campos descritos na Tabela IV.

TABELA IV

REGISTRE A ENTRADA EM UMA LINHA DO TEMPO DE CHAVES SOBERANAS



Uma solicitação de adição ao TimeLine deve conter dados relacionados ao SK e ao serviço (ou serviços) a quem será relacionado. O campo “Chave Soberana” contém o nome de domínio (“nome”) ao qual o SK se aplica. Se o “Wildcard” não estiver definido [100], um SK diferente pode ser usado por subdomínios; caso contrário, apenas o SK será válido para cada subdomínio.

Em relação ao tipo de chave, expresso com “tipo de chave”, a especificação [100] propõe o uso de ECC para um armazenamento e compressão ótimos [101]. O par de chave SK deve ser gerado pelo solicitante. A chave pública juntamente com um comprovante de posse da chave privada devem ser adicionados à solicitação. O campo “Chave pública soberana” contém a chave pública do SK.

A estrutura permite limitar os serviços aos quais o SK se aplica por meio do campo “protocolos”. O campo contém uma cadeia de texto com as rotas alternativas para os serviços delimitadas por ponto e vírgula. Por exemplo, HTTP na porta 8080 ou um endereço de roteamento onion [100]. Por fim, essa entrada da tabela também contém a data de vencimento no campo “expira em”.

O campo “nomes herdados” permite que os SKs sejam reemitidos para o domínio se os SKs atuais forem comprometidos ou revogados. O valor é uma lista de domínios com permissão para solicitar a adição de uma nova entrada nessas circunstâncias. Na verdade, é uma delegação para registrar novamente o domínio se, e somente se, o SK for revogado. Dessa forma, o SK evita que os proprietários de domínio listados no campo alterem outros SKs se seus domínios estiverem comprometidos, mas não o SK original.

Solicitar a inclusão de um SK em um domínio exige o fornecimento de evidências de que o domínio está sob o controle do solicitante. Essa evidência pode ser um certificado assinado por uma CA confiável contendo o nome de domínio em uma extensão subjectAltNames ou uma resposta DANE DNSSEC (consulte a Seção IV-F2). Provas obtidas durante a adição de um SK que os clientes podem verificar mais tarde

O servidor que mantém o TimeLine deve realizar uma verificação OCSP [102] antes de adicionar o SK. A verificação OCSP não é adicionada ao registro devido a restrições de espaço.

Para garantir a consistência da solicitação e evitar manipulações de entidades intermediárias durante o processo de solicitação ou do servidor TimeLine após a solicitação, a solicitação é assinada com a chave privada do SK e o resultado adicionado ao campo “Claim\_signature”. Essa assinatura também serve como prova de posse.

Apesar de não estar diretamente declarado na especificação [100], entende-se que os campos contidos em “Chave soberana”, “Em caso de revogação” e “evidências para reivindicação” são protegidos pela assinatura em “Claim\_signature”. Qualquer alteração posterior executada pelo solicitante posteriormente é adicionada como um novo registro ao histórico preservado pelo TimeLine.

A assinatura do servidor TimeLine sobre todos os campos da Tabela IV é realizada com uma chave privada pertencente ao servidor TimeLine e pode ser realizada offline. Assim, para garantir o recurso somente de acréscimo do TimeLine, um número de série crescente e um carimbo de hora são adicionados a cada entrada.

A estrutura TimeLine também pode incorporar outras entradas:

• Referências a outros TimeLines. É possível que as entidades que gerenciam um TimeLine obtenham a chave do TimeLine comprometida, portanto, as solicitações de adição de SK podem ser enviadas para vários TimeLines. O campo de registro “Incorporar por referência” permite vincular um registro em um TimeLine com outros registros em diferentes TimeLines correspondentes à mesma operação.

• Revocations fornece um mecanismo para o proprietário do SK revogar a chave. Uma revogação efetiva requer a data de revogação, o nome do SK (nome de domínio) e uma assinatura sobre os parâmetros. As informações de revogação serão adicionadas ao TimeLine com um número de série exclusivo e um carimbo de data / hora.

• A reemissão de um SK revogado permite que um SK revogado seja reemitido mediante uma evidência fornecida a partir de um domínio listado no campo “Em caso de revogação”. A estrutura será adicionada ao TimeLine com um número de série e um carimbo de hora.

• As alterações de protocolo rastreiam as alterações no campo "Protocolos"

• Alterações na lista de CA raiz. Cada entidade gerenciando um TimeLine deve manter e manter uma lista de CAs confiáveis. A lista pode ser modificada e cada mudança na lista é publicada no TimeLine com um número de série e um carimbo de hora. Da mesma forma, a Transparência de certificado, descrita na Seção IV-A, certificados autoassinados, CAs de domínio e ilhas de segurança não podem ser usados com SK, apesar de serem confiáveis por meio do DANE-EE e do DANE-TA descritos na Seção IV-F2.

SKs definem três entidades que são servidores, espelhos e clientes TimeLine. Os servidores TimeLine gerenciam e custodiam os TimeLines. Ao contrário de CT, SK especifica que o conjunto de servidores TimeLine para a Internet atual e previsível deve ter N entradas com N entre 10 e 30. Também é declarado que os servidores TimeLine devem ser escolhidos para garantir a diversidade de jurisdição, filosofia operacional ou políticas de segurança, portanto, o serviço estará disponível mesmo se vários servidores TimeLine estiverem comprometidos ou desativados.

Cada TimeLine Server deve ter um par de chaves para a assinatura do TimeLime conforme explicado anteriormente. De acordo com a especificação SK, a lista de servidores TimeLine e suas chaves públicas correspondentes devem ser distribuídas junto com o software, como acontece com as listas de CA raiz confiável de PKI.

Os “espelhos” aumentam o desempenho e a disponibilidade geral do sistema, mantendo cópias atualizadas dos servidores TimeLine. Os “espelhos” devem ser identificados por um endereço IP, porta e chave pública privada, introduzindo uma lista adicional de chaves.

SK não define verificações de consistência com base na estrutura como o CT, mas em nível de protocolo. Os servidores TimeLine devem responder aos espelhos e clientes com todas as entradas desde um determinado número de série S. Cada registro retornado por um servidor TimeLine deve ser acompanhado por uma mensagem de atualização chamada "Timeline Freshness Message" (TFM) com os campos: "TimeStamp" do solicitação; “Maior número de série até o momento” com o maior número de série de registros SK disponíveis no TimeLine Server no momento da solicitação; “Número de série de atualização de CA mais alto até a data” o número de série mais alto do registro de atualização de CA e a assinatura do TimeLine.

A especificação descreve a avaliação do frescor considerando as respostas com menos de 24 horas como recentes, respostas com mais de 24 horas e menos de 48 horas como aceitáveis, aquelas com mais de 48 horas, mas com menos de 2 semanas como inaceitáveis e com mais de 2 semanas como fatais.

Para verificar o funcionamento de um TimeLine Server, o verificador precisa verificar o TFM das respostas, determinando que um TimeLine Server falhe em sua função se, observando o TimeLine:

• duas entradas diferentes no TimeLine com o mesmo número de série

• duas entradas diferentes com discrepâncias no carimbo de data / hora

• um TFM com um carimbo de data / hora precedente e um número de série maior do que um TFM observado anteriormente

• e entrada para um SK com uma assinatura inválida

Se o verificador, um espelho ou um cliente detectar uma falha em um servidor TimeLine, ele deve manter uma cópia das entradas, desconfiar do servidor TimeLine com falha e adicionar o servidor TimeLine a um registro de servidor TimeLine inválido que pode ser aprendido por outros espelhos e clientes.

Para sincronizar a lista de servidores TimeLine inválidos entre as partes, o protocolo SK usa um campo denominado “renegation\_traking” de 32 bits, que contém os bytes menos significativos do hash da lista de servidores inválidos. Se um Cliente receber uma mensagem cujo campo “renegation\_traking” seja inconsistente com o que mantém, inicia um processo de sincronização que requer a troca de suas listas.

Considerações de segurança para o cliente: a especificação SK [100] não afirma claramente como o domínio de destino de um certificado adicionado ao TimeLine é verificado, enquanto outras propostas o fazem, como DANE.

Se a assinatura do registro for realizada offline, não há indicação do tempo que leva ao contrário do CT que promete incorporar o registro antes do MMD.

Como já foi discutido antes, cada servidor TimeLine deve ter um par de chaves para a assinatura TimeLine. Esse par de chaves deve ser distribuído para espelhos e clientes com o software. Além disso, os espelhos são identificados por IP, porta e chave pública, portanto, um conjunto adicional de chaves é introduzido no sistema.

A especificação não indica se o TimeStamp do TFM corresponde à hora da solicitação ou à hora do último registro. No primeiro caso, o servidor deve assinar imediatamente após a solicitação de Mirror, levando a um ataque de amplificação que pode ser perigoso se o número de solicitações aumentar. No segundo caso, o servidor não deve precisar assinar imediatamente, mas, a menos que o canal entre o servidor e o cliente esteja protegido, uma entidade no meio pode usar respostas anteriores do TimeLine Server para mascarar atualizações recentes.

Em relação à privacidade, os espelhos podem aprender o endereço IP dos clientes que verificam os certificados. A especificação propõe duas alternativas. A primeira é que os clientes devem usar espelhos gerenciados por seus ISPs. Desta forma, apesar de o ISP poder conhecer os domínios que um determinado Cliente visita, já o pode fazer através dos servidores DNS.

A segunda consiste em um mecanismo de encaminhamento iniciado pelo cliente. É proposto que todo Mirror tenha duas portas, a principal (443) para enviar a solicitação ao Mirror e uma segunda, chamada “SK mirror port” ou MP, que é usada para receber e encaminhar respostas. Desta forma, os clientes podem usar um espelho como proxy, melhorando a privacidade. No entanto, a segunda proposta não protege os usuários de espelhos cooperativos.

Finalmente, a especificação afirma que um cliente solicita informações aos espelhos a cada 24 horas. Apesar de reduzir a carga dos Mirrors, ele abre uma janela de ataque de 24 horas.

C. Asserção de confiança para chaves de certificado

Trust Assertion for Certificate Keys (TACK) [103] permite aos usuários associar um domínio com um certificado usando uma estrutura chamada TACK, assinada com uma “chave de assinatura TACK” ou TSK, que é escolhida pelo dono do domínio. Os TSKs são confiáveis para os clientes e não devem ser alterados com frequência, enquanto as alterações nos certificados do servidor não são limitadas em frequência. TACK constitui, na prática, uma mudança no modelo de confiança de PKI à medida que move a confiança da lista de CA raiz para os TSKs. Além disso, ele propõe um mecanismo de revogação para certificados TLS comprometidos, enquanto um mecanismo de sobreposição para atualizar TSKs em que o antigo e o novo TSK coexistem durante um tempo até que o antigo seja finalmente desativado.

Ao contrário das propostas acima mencionadas, o TACK não apresenta uma estrutura global verificável. TSKs e TACKs são gerados e entregues por servidores para que o cliente possa processá-los e mantê-los para estabelecer uma relação de confiança de longo prazo com um servidor

O sistema define dois ciclos de vida diferentes, um para TACKs e outro para TSKs. O ciclo de vida do TACK-TSK é o seguinte:

• Geração de TSK: o servidor gera uma chave ECDSA [104] que seria usada para assinar um ou mais TACKs de nomes de domínio.

• Criação de TACK: o TACK contém metadados para associar um certificado de servidor a um TSK. Uma vez gerado, o TACK é assinado com o TSK.

• Implementação TACK: um TACK vinculando o certificado TLS e o TSK é fornecido a cada servidor no domínio a ser protegido. Esses servidores anunciam o TACK configurando o “flag de ativação”.

• Geração de TACK: quando o TACK expira ou o servidor altera o certificado TLS, um novo TACK é gerado.

• Revogação de TACKs: caso um certificado TLS seja comprometido, um novo TACK pode ser criado incrementando o campo “min generation”.

• Desativação de TACKs: o dono do servidor pode desativar um TACK removendo o “flag de ativação”, para que os servidores possam removê-lo após um período de 30 dias.

• Sobreposição de TACKs: quando um TSK vinculando um certificado TLS a um determinado servidor por meio de um TACK precisa ser alterado, o servidor publica um novo TACK assinado com o novo TSK. O novo TACK é distribuído junto com o antigo TACK durante um período de tempo, para que os clientes possam ativar a nova associação enquanto o serviço não é interrompido.

Os campos em um TACK entregue pelo servidor ao cliente durante um handshake TLS são os seguintes:

• public\_key: contém os inteiros correspondentes a um ponto na curva elíptica p-256 [105] que representa a chave pública da TSK que assinou a TACK.

• min\_generation: contém o valor correspondente ao campo min\_generation do TSK associado ao TACK.

• geração: sinaliza a geração do TACK de forma que todo outro TACK cuja geração seja menor que o máximo min\_generation do TSK de assinatura é considerado revogado.

• expiration: data após a TACK ser considerada expirada.

• target\_hash: um hash SHA256 [106] da chave pública [24] do certificado TLS usado pelo servidor TLS.

• assinatura: uma assinatura ECDSA usando o TSK em todos os campos anteriores.

As associações ou “pins” entre TSKs e nomes de domínio, representados por TACKs, são organizados em repositórios ou “lojas”. Cada loja mantém um mapa que relaciona os nomes de domínio totalmente qualificados (FQDNs) com um ou mais conjuntos de atributos. Dentre esses atributos estão a data de emissão, a data de vencimento, a chave pública TSK (ou seu hash) e o campo min\_generation, que deve ser igual para cada TACK assinado pelo mesmo TSK. Um cliente pode ter uma ou mais lojas que podem ser locais (opcional) ou fornecidas por uma parte remota e pode compartilhar pinos com outros clientes. O protocolo permite que os clientes baixem TACKs de outros e publiquem TACKs descobertos usando um terceiro confiável.

Os TACKs são entregues ao cliente usando uma extensão TLS, para que os clientes possam verificar se estão se conectando ao servidor apropriado, corroborando o TACK recebido com as informações contidas nas lojas. Quando um servidor compatível recebe uma mensagem de handshake TLS, ele negocia o uso da extensão TACK TLS e entrega uma TACK ao cliente. Se o cliente aprendeu a mesma TACK várias vezes, ele pode criar um “pino” entre o nome de domínio e o TSK em uma de suas lojas. A validade do “pino” é igual ao período de tempo em que a relação foi observada, limitando o impacto de “pinos” errôneos ou maliciosos. Os processos de troca e verificação vêm a seguir.

O cliente verifica o handshake TLS com extensão TACK como válido se o handshake resultar em um canal criptografado, se a extensão TACK estiver presente e se o TACK entregue for válido. Uma TACK é considerada válida se o campo “generation” for maior que o campo “min\_generation” (do TSK), “expiration” é datado no futuro, “target\_hash” está correto e a assinatura verifica. Uma vez que o TACK é verificado, o cliente procura um “pin” em seus armazenamentos para o nome de domínio do servidor. Se o TACK armazenado for igual ao recebido, a conexão é marcada como “Confirmada”, caso contrário, como “Contradita”. Se não houver nenhuma associação registrada para o nome de domínio do servidor, a conexão será marcada como “liberada”. No caso de “desafixado”, se o servidor TACK foi observado antes, o cliente pode ativar um “pin” e criar uma associação com uma data de término: fim = atual + MIN (30 dias, atual - inicial), sendo “ inicial ”a data da observação inicial.

Considerações de segurança para o cliente: Como foi mencionado, um cliente pode usar uma ou mais lojas, uma local e outra remota. Os remotos podem ser fornecidos por terceiros, mas nem as organizações que gerenciam os repositórios nem os requisitos para que um terceiro se torne confiável são especificados. Além disso, se os repositórios contendo TACKs relacionados a uma coleção de servidores de uma determinada área geográfica pertencerem a uma operadora ou fornecerem TACKs por tópico, as consultas do cliente podem revelar interesse ou hábitos que aumentam as preocupações com a privacidade.

A especificação também afirma que os clientes podem compartilhar “pinos” com outros clientes e até mesmo publicar aqueles que descobriram usando um terceiro confiável ou serviço de compartilhamento. Isso levanta questões de privacidade, pois outras entidades podem aprender os hábitos e interesses do usuário. Além disso, apenas TACKs válidos podem ser compartilhados, pois podem ser facilmente verificados, portanto, os clientes não podem adicionar perturbações aos TACKs observados para evitar a criação de perfil. Finalmente, isso requer que o cliente suporte um protocolo de compartilhamento que pode exigir armazenamento adicional e poder de processamento

D. Autorização da Autoridade de Certificação DNS

A Autorização de Autoridade de Certificação DNS (CAA) permite que os proprietários de domínio especifiquem quais CAs podem emitir certificados para seus nomes de domínio. Ao contrário do CT ou SK, a proposta não define uma estrutura verificável para armazenar evidências de emissão de certificados, nem um mecanismo de verificação cruzada. Ele apenas fornece um mecanismo para que as CAs verifiquem se têm permissão para emitir um certificado para um determinado domínio ao receber uma Solicitação de Assinatura de Certificado (CSR). Assim, o CAA intervém apenas antes de emitir o certificado.

A CAA define um registro DNS denominado “Registro de recursos DNS de Autorização de Autoridade de Certificação (CAA)” que permite ao proprietário do domínio, ou à entidade que gerencia o DNS primário desse domínio, especificar uma lista de CAs autorizados com o propósito de emitir um certificado. Dessa forma, qualquer CA compatível deve consultar o servidor DNS dos proprietários do domínio em busca de um registro CAA a fim de verificar se ele está autorizado para esse domínio. Desta forma, o risco de emissão errônea inadvertida é reduzido.

De maneira semelhante aos registros TLSA DANE [107], discutidos na Seção IV-F, o CAA está sob o controle do proprietário do domínio e nenhum terceiro está envolvido. No entanto, a diferença fundamental é que o CAA ajuda o CA a determinar se está autorizado antes de emitir o certificado, enquanto o TLSA permite que os clientes verifiquem se um certificado de servidor, usado em TLS, está autorizado pelo proprietário do domínio para fins de autenticação de uma conexão criptografada .

De acordo com a especificação, a conformidade com o registro CAA DNS é necessária, mas não condição suficiente, para a emissão do certificado, uma vez que as solicitações de certificado também devem estar de acordo com a “Política de Certificado” da CA. Como parte desse critério, é necessário que as CAs publiquem sua “Declaração de Práticas de Certificados” (CPS) e contem com um processo de auditoria externa. O CAA não busca a criação de ilhas de segurança com CAs locais, mas impõe que qualquer certificado deve ser emitido por um CA no registro CAA. O CAA no registro deve ser um CA confiável para o cliente, portanto, o CA deve estar na lista de CA confiáveis do cliente. Desta forma, se um CA listado em um registro CAA para um domínio não é confiável para um determinado cliente, mas emitiu um certificado de servidor TLS, esse cliente não pode confiar no certificado de servidor TLS independentemente do registro CAA.

Os registros CAA não podem ajudar os clientes na verificação cruzada, uma vez que um certificado de servidor TLS pode ter uma vida longa (cerca de anos), enquanto o proprietário do domínio pode alterar seu CA de fornecimento e, portanto, o registro CAA, várias vezes durante a vida dos certificados TLS.

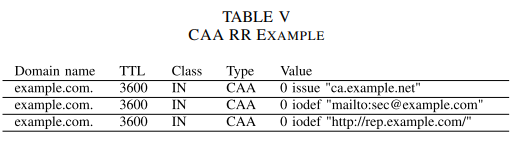
Um CAA RR consiste em um conjunto de sinalizadores e pares de valor de rótulo conhecidos como propriedades. Várias propriedades diferentes podem ser associadas a um domínio publicando diferentes CAA RR sob o mesmo DNS de domínio. Uma propriedade pode ser sinalizada como “crítica para o emissor”, indicando que deve ser interpretada corretamente pelo emissor (CA) antes de emitir ou, de outra forma, desistir de emitir.

As propriedades CAA mais representativas são:

• issue <Issuer Domain Name>: autorizar o proprietário do domínio “Nome de Domínio do Emissor” a emitir certificados para nomes de domínio gerenciados pelo DNS consultado.

• issuewild < Issuer Domain Name >: igual ao anterior, mas permite nomes de domínio com caracteres curinga.

• iodef <URL>: define a url para a qual as tentativas de solicitação de certificado inconsistentes devem ser relatadas. Usa o formato IODEF [108]



A Tabela V mostra um exemplo em que o nome de domínio example.com afirma que a única CA autorizada para esse domínio é ca.example.net; solicitações inconsistentes devem ser relatadas por e-mail e um URL.

Apesar da especificação recomendar que os registros CAA sejam autenticados com DNSSEC, isso não é obrigatório. Assim, seria possível que um invasor descartar, alterar ou inserir registros CAA fraudulentos se o DNSSEC não for usado.

O CAA não impede a falsificação de identidade ou o uso fraudulento de certificados, portanto, contribui para a “fixação de certificados”, mas não é um mecanismo por si só. As considerações de segurança para o cliente não podem ser discutidas, pois nenhum cliente está envolvido no CAA.

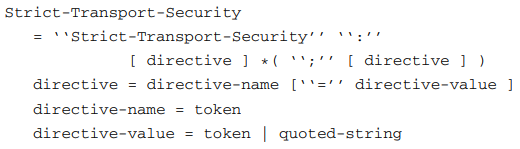
E. HTTP Strict Transport Security e HTTP Protocolo de Pinagem de Chave Pública

HTTP Strict Transport Security (HSTS) [109] descreve um mecanismo que permite que sites que usam HTTP declarem que são acessíveis apenas por meio de uma conexão criptografada, como HTTPS. Assim, a especificação é limitada a HTTP que pode ser usado sobre TLS [110] com o esquema URI “https”.

O HSTS é baseado em pesquisas anteriores [111], [112] enfrentando ameaças de atacantes passivos e ativos. No primeiro grupo de ataques, um invasor escuta na rede os cookies de sessão que, apesar de entregues em primeiro lugar por meio de um canal protegido (HTTPS), são entregues em texto não criptografado (HTTP) quando o cliente carrega outros recursos.

No segundo grupo de ataques, um invasor pode usar servidores DNS envenenados ou modificar frames desprotegidos para obter as informações da sessão. Em seguida, o tráfego pode ser redirecionado para servidores Web desprotegidos ou para servidores usando certificados autoassinados, uma vez que, conforme discutido na Seção III, os clientes “clicarão” após um aviso. Outras ameaças, como phishing ou malware, não são tratadas pelo HSTS.

O HSTS define um cabeçalho HTTP que deve ser entregue usando HTTP sobre TLS para o cliente (Agente do Usuário ou UA) com o formato a seguir [113].



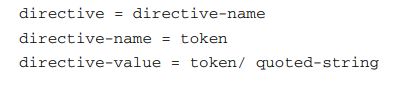
Dentre as diretivas incluídas no cabeçalho, o UA deve lembrar a política do servidor (“max-age”), se subdomínios devem ser tratados da mesma forma (“includeSubDomains”) ou se devem ser adicionados à lista de permanentes Servidores HTTPs (“pré-carregamento”). Os clientes devem armazenar e manter as políticas HSTS no UA. Alguns navegadores incluem alguns servidores pré-carregados, evitando que os usuários os acessem sem TLS. No entanto, o HSTS não evita problemas derivados de certificados mal-intencionados emitidos por CAs comprometidos, mas evita que as sessões sejam redirecionadas para protocolos inseguros e, portanto, evita que as informações da sessão sejam roubadas.

O protocolo HTTP Public Key Pinning (HPKP) [114] permite que o cliente detecte quando uma cadeia de certificados ou confiança mudou inesperadamente. HPKP define um cabeçalho HTTP que permite ao UA aprender quais estruturas SubjectPublickKey devem estar presentes na cadeia de certificados em futuras conexões TLS com o mesmo servidor. O objetivo é evitar ataques MITM com base em certificados comprometidos e deve ser usado em conjunto com o HSTS.

Assim, HPKP define uma relação entre um nome de domínio e uma cadeia de certificados. O mecanismo proposto segue um “trust-on first-use” (TOFU). A primeira vez que o cliente acessa o servidor não tem conhecimento sobre o servidor. Portanto, não seria capaz de detectar um ataque MITM. No entanto, nesta primeira conexão, o cliente (UA) aprende a cadeia de certificados válidos para aquele servidor evitando ataques futuros.

O cabeçalho HTTP HPKP deve ser entregue por TLS a um cliente (UA) com o seguinte formato (no qual os campos “token” e “string entre aspas” são formatados de acordo com [113]).





A “Diretiva de Pin” permite que o servidor Web indique os certificados que devem ser associados ao host. Para fazer isso, ele fornece uma sequência de hashes SHA256 das estruturas SubjectPublickKey da cadeia de certificados esperada. A diretiva “max-age” especifica por quanto tempo (em segundos) um cliente deve se lembrar da cadeia, considerando o servidor como um “Host Pinned Conhecido”. Assim como o HSTS, ele também inclui uma diretiva chamada “includeSubDomains” e um “report-uri” para relatar uma verificação insatisfatória.

Considerações de segurança para o cliente: HPKP define um mecanismo conhecido como TOFU (Trust-on-first-use), pois na primeira vez que o cliente se conecta ao servidor, ele não possui as informações necessárias para validar a associação (pin), de forma que não seria capaz de detectar um ataque MITM. Além disso, como os usuários comuns desconsideram os avisos do navegador, também é viável um ataque MITM mesmo para “Host Pinned Conhecido”.

HSTS e HPKP são projetados apenas para HTTP. Além disso, os clientes devem armazenar associações HPKP e lembrar as políticas de HSTS.

F. Autenticação baseada em DNS de entidades nomeadas

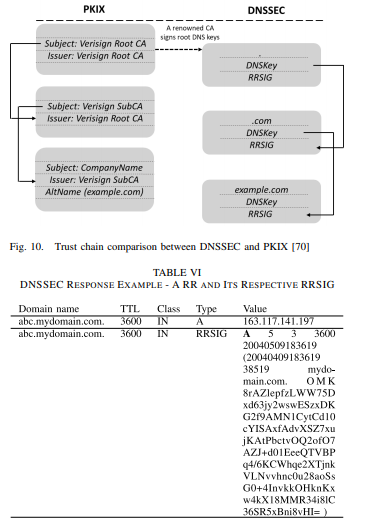
Conforme discutido na Seção III, a PKI não tem uma única autoridade raiz capaz de verificar todos os certificados. O DNS tem um único domínio raiz (.) Sob o qual cada árvore de domínio cresce formando uma árvore (.com., .Net. E outros). Assim, ao contrário da PKI em que cada CA confiável pode agir impunemente emitindo certificados, no DNS apenas a raiz pode agir impunemente alterando a base de dados de nomes de domínio.

Ramificações ou subdomínios no DNS são delegados a seus respectivos proprietários de domínio. Devido a isso, o DNS tem uma infraestrutura pronta para fornecer um mecanismo de pinning de certificado adequado reutilizando a infraestrutura DNS com vários benefícios claros: não requer a distribuição de novas credenciais, uma vez que o DNS já está implantado e pode fornecer registros autenticados usando extensões de segurança ( DNSSEC); não requer a criação de novos serviços e permite que os proprietários de domínios gerenciem suas relações de confiança por conta própria, sem a intervenção de terceiros

A Seção IV-F1 descreve resumidamente as extensões de segurança do DNS para uma melhor compreensão da Autenticação de Entidades Nomeadas baseada em DNS [107] (DANE) que será descrita na Seção IV-F2.

1) Introdução ao DNSSEC: Extensões de Segurança do Sistema de Nomes de Domínio (DNSSEC-bis) [115], [116] pertencem a um conjunto de especificações da IETF que permite aos clientes DNS (resolvedores) autenticar a fonte das respostas DNS, autenticar as respostas de domínio inexistentes (evitando certos ataques) e verificar a integridade das respostas DNS. Ele não fornece confidencialidade a menos que usado em conjunto com TLS ou DTLS [117], [118].

DNSSEC [119] foi proposto pela primeira vez em 1997, mas a primeira versão tinha grandes problemas de escalabilidade, uma vez que as zonas pai devem assinar registros sobre as alterações em cada ramificação delegada ou zona filha. A proposta atual, conhecida como DNSSEC-bis, propõe uma indireção na assinatura usando registros chamados “Registros de recursos de signatário de delegação (DS)” que melhoram a escalabilidade.



Cada registro DNS no DNSSEC é entregue junto com um registro adicional conhecido como RRSIG, que contém uma assinatura sobre o registro original usando uma chave de assinatura de zona (ZSK). Os registros RRSIG permitem autenticar as informações do DNS. O ZSK é certificado pela Key Signing Key (KSK), uma chave de longo prazo. KSKs e ZSKs são locais para suas respectivas zonas DNS, portanto, para criar uma cadeia de confiança para o pai, um DNSSEC pai usa registros DS para indicar os KSKs para as zonas filho. Este processo de delegação de assinatura, coerente com a delegação de nome de domínio DNS, é seguido até a raiz

A própria autoridade raiz do DNS tem sua própria declaração de prática e uma cerimônia complexa para lançar um novo RootZone KSK.2 Essa cerimônia não sendo alterada é a âncora de confiança no DNSSEC. A sequência de registros DS da raiz a qualquer folha na árvore DNS leva a uma cadeia de confiança alternativa à cadeia de confiança de PKI. A Fig. 10 mostra uma comparação entre as duas cadeias de certificados (o leitor deve observar que a raiz do DNS é única, ao passo que a PKI não é e depende de um determinado certificado).

A Tabela VI mostra um exemplo de registro DNSSEC que é discutido a seguir.

O valor de um RRSIG é composto por vários campos. O primeiro indica o tipo de registro que está sendo assinado (A no exemplo). O próximo campo indica o algoritmo usado para criar a assinatura (5 RSA / SHA1 no exemplo). Então, o RR contém os registros ou rótulos, que são usados para validar os registros gerados a partir de um curinga. No exemplo, tem o valor 3 assim, o original é composto por 3 rótulos. abc.mydomain.com é o original, pois abc.mydomain.com tem 3 rótulos (abc, mydomain, com), portanto, não foi gerado usando um caractere curinga.

Se 2 fosse usado em vez de 3, o campo mostraria que o nome de domínio foi gerado a partir de \* .mydomain.com e que deve ser usado para a verificação de acordo com a especificação [116]. Em seguida, o RR TTL original é incluído (3600 no exemplo) evitando que as chaves de assinatura comprometidas antigas sejam usadas após o TTL. Após o TTL, o registro inclui a data de expiração (20040509183619) e a data de início (20040409183619), a chave usada para identificar a chave de assinatura (38519), bem como o nome da chave de assinatura (meudomínio). Vários RRs podem usar os mesmos RRSIGs, o conjunto de RRs que compartilham o RRSIG são conhecidos como RRSet. Finalmente, a assinatura do RRSet, é gerada da seguinte forma [115]:

signature = sign(RRSIG\_RDATA | RR(1) | RR(2)... )

onde | significa concatenação; RRDATA são os campos do registro RRSIG, incluindo o nome da zona canônica e excluindo o campo de assinatura; conforme mencionado, todo RR pertencente ao RRSIG contribui com RR (i), que é gerado da seguinte forma:

RR(i) = owner | type | class | TTL |

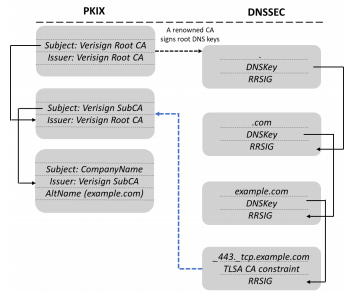
RDATA length | RDATA

Cada RR que contribui para a assinatura deve pertencer à mesma zona de assinatura, manter o TTL original e ter a mesma classe / tipo.

A combinação de proprietário, classe e tipo pode ser usada para determinar que o RRSIG no exemplo autentica A RRs de abc.mydomain.com e nenhum curinga foi usado. Assim, a assinatura pode ser verificada usando uma DNSKEY da zona mydomain.com com a chave 38519 usando o algoritmo 5.

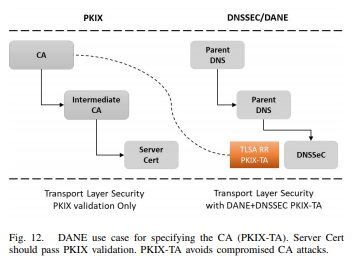
2) Autenticação baseada em DNS de entidades nomeadas (DANE): A autenticação baseada em DNS de entidades nomeadas [107] (DANE) depende do DNSSEC para autenticar os registros DNS DANE com o propósito de associar nomes de domínio com credenciais (certificados PKI). Como foi discutido na Seção III, o uso fraudulento de certificados está entre os problemas mais preocupantes de PKI quando usado com TLS. O problema é que qualquer CA comprometida pode emitir certificados para sites conhecidos que passarão na validação de PKI e servirão para o propósito do ataque, sem o conhecimento do proprietário do domínio. Conforme as propostas mencionadas, o DANE também busca criar uma espécie de associação entre nomes de domínio e certificados (“Certificate Pinning”) resistente a esses ataques.

O DANE permite que os proprietários de domínio incluam informações sobre as credenciais de autenticação de seus serviços permanentes em seu DNS. Considerando que o DNS é normalmente consultado para resolução de nomes pelo cliente imediatamente antes de se conectar ao servidor, o cliente pode receber informações relativas à credencial que deve ser recebida do servidor durante um handshake de TLS antes de realmente se conectar ao servidor. Assim, evita que certificados maliciosos sejam usados para personificar servidores gerenciados pelo proprietário do domínio (consulte a Figura 11).

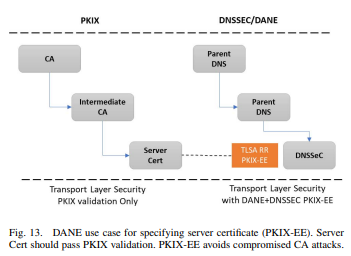
  
Fig. 11. Caso de uso DANE: especificando o CA que deve ser usado por um serviço (PKIX-TA). A figura mostra que o registro DANE pode indicar que a CA que deveria ter emitido o certificado do servidor é “Verisign SubCA”, portanto, o registro TLSA restringe a CA. As linhas sólidas representam a hierarquia em DNSEC / DANE e PKI. As linhas tracejadas indicam a relação entre as entidades.

O DANE define um novo registro DNS chamado TLSA que permite a uma zona DNS declarar como os clientes, resolvendo nomes de domínio dessa zona, devem processar os certificados recebidos por meio de uma conexão TLS. As especificações DANE [RFC 6698], [120] definem os seguintes casos de uso, dependendo dos parâmetros do registro:

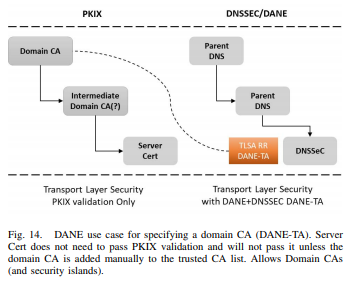
• Restrição CA: o registro especifica o certificado, ou a chave pública do certificado da CA, que deveria ter emitido o certificado TLS. Isso não afeta a maneira como o cliente lida com a confiança, uma vez que o certificado TLS deve passar na validação de PKI. Ele apenas especifica exatamente qual CA deve ter emitido o certificado do servidor, e por esse motivo é chamado de “restrição de CA” (PKIX-TA [121]). Veja a Fig. 12.



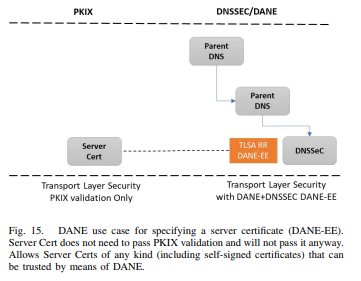
• Limitação do certificado do servidor: o registro contém o certificado (ou a chave pública) do certificado TLS usado pelo servidor. É chamado de “restrição de certificado de serviço” (PKIX-EE [121]), pois define o certificado que o servidor deve usar. Apesar do registro especificar um certificado concreto, o certificado deve passar na validação de PKI. Veja a Fig. 13.



• Especificação da CA: o registro contém o certificado (ou a chave pública) da CA que deveria ter emitido o certificado usado pelo servidor. Ao contrário do primeiro caso de uso, não é necessário que o certificado passe na validação de PKI. Neste caso, o DANE modifica a forma como o cliente gerencia os trustes e por isso é chamado de “asserção de âncora de confiança” (DANE-TA [121]). Ele altera a confiança do cliente, pois especifica uma CA concreta que deve ser confiável, mesmo se a CA não for confiável para o cliente. Este caso de uso oferece suporte a “ilhas de segurança” dentro do domínio para que os domínios possam emitir seus próprios certificados usando sua própria CA. Veja a Fig. 14.



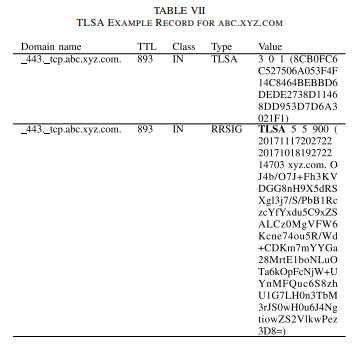
• Especificação do certificado do servidor: o registro contém o certificado (ou a chave pública) do certificado TLS usado pelo servidor. Este caso de uso, denominado “certificado emitido por domínio” (DANE-EE [121]), permite que um domínio emita seus próprios certificados sem um terceiro ou CA. Da mesma forma que a "declaração de âncora de confiança", não requer validação de PKI alterando a confiança do cliente. Na verdade, esse caso de uso permite que certificados autoassinados e efêmeros sejam emitidos por um domínio e confiáveis para o cliente. Veja a Fig. 15.



O registro TLSA define um “seletor” que sinaliza se o conteúdo dos registros se refere a um “Certificado Completo” [24] (Cert) ou à sua chave pública, “SubjectPublicKeyInfo” [24] (SPKI). Além disso, é possível indicar se o registro se refere ao conteúdo exato ou ao seu hash por meio do campo “matching”. Finalmente, o campo “dados de associação de certificado”, definido pelos dois campos anteriores, define o objeto alvo da associação como bruto (completo), com hash SHA-256 ou SHA-512 para certificado ou chave pública.

A Tabela VII mostra um registro de exemplo TLSA retornado por um DNS afirmando o servidor abc.xyz.com com uma interface REST [15] rodando na porta 443, deve usar um certificado emitido por uma CA concreta.

O exemplo na Tabela VII contém um registro TLSA DANE e seu registro DNSSEC RRSIG relacionado. O TLSA indica que o servidor abc.xyz.com deve usar um certificado (3 - "certificado emitido por domínio") cuja estrutura binária completa (0 - "Certificado completo") deve ter um hash SHA-256 (1 - SHA-256) com um valor igual ao fornecido.



Para evitar manipulação e autenticar os registros TLSA, o registro TLSA deve ser entregue junto com seu registro RRSIG correspondente. O registro RRSIG no exemplo protege um TLSA RR, usando o algoritmo 5 (RSA / SHA1) [122]. Uma proteção mais forte é obtida com RSA / SHA-256 ou RSA / SHA-512 (IETF RFC 5702), e ECDSA / SHA-256 (IETF RFC 6605). O registro RRSIG se refere a um nome de domínio com 3 rótulos com um TTL de 900. Ele também contém as datas de / para válidas, o tag com o valor 14703 e a zona à qual o registro pertence e de onde a chave pública de assinatura deve ser recuperado para verificar o registro.

Considerações de segurança para o cliente: DANE especifica que o cliente TLS compatível deve suportar os três campos de registros TLSA: uso do certificado, seletor e tipo de correspondência. Os clientes devem oferecer suporte aos quatro valores de uso de certificado (casos de uso) possíveis em registros TLSA e também usar os tipos de seletor 0 (certificado completo) e 1 (chave pública) para comparar uma associação de certificado com um certificado do handshake TLS usando hashes ( SHA256 ou 512) ou diretamente conforme especificado pelo campo de tipo de correspondência do registro TLSA.

**Seção 5 (COMPARAÇÃO DE SOLUÇÕES DE FIXAÇÃO DE CERTIFICADOS PARA IOT / M2M)**

Esta seção contém uma comparação de vários aspectos das soluções de pinagem de certificado discutidas em mais detalhes do que outras discussões, comparando mais soluções e com um foco particular em sua aplicação para cenários IoT / M2M. A Tabela VIII, ao final da seção, contém um resumo dos aspectos mais importantes da comparação.

A. Use of Side Channels - Uso de canais laterais

Os canais laterais permitem realizar algumas das validações necessárias, mas não usam o protocolo original. Por exemplo, o uso de TLS pode acionar a verificação, mas ela será realizada usando um protocolo diferente ou uma instância diferente do mesmo protocolo.

O impacto do uso de canais laterais depende de quando o canal lateral será usado. O primeiro caso consiste na utilização do canal lateral durante o handshake TLS. A segunda corresponde à utilização do canal lateral em um momento diferente, gerando dados que podem ser utilizados em um futuro handshake TLS.

O CT não requer o uso de canais laterais durante o handshake TLS, mas propõe que os auditores, no lado do cliente, contrastem as informações com os monitores de forma assíncrona e, opcionalmente, compartilhem os SCTs observados com outros clientes por meio de um protocolo de fofoca. Apesar do uso de canais laterais durante o handshake não ser obrigatório, é possível verificar os SCTs de forma síncrona. Esta verificação síncrona pode ser necessária em IoT e M2M, uma vez que os dispositivos podem suspender sua atividade durante longos períodos de tempo, dificultando a verificação assíncrona e o uso de protocolo de fofoca. Em relação às dificuldades com o protocolo de fofoca, isso não significa que um dispositivo IoT não possa cooperar, mas a eficácia da cooperação deve ser contextualizada. Um dispositivo IoT / M2M pode ser limitado em capacidade de armazenamento, afetando o armazenamento de SCTs observados. Além disso, deve-se levar em consideração o balanço energético entre o gasto de energia nas transmissões necessárias para cumprir a finalidade do dispositivo e aquelas motivadas pelo protocolo CT.

SK requer o uso de canais laterais para verificar o pino. O servidor entrega o pin (estrutura de dados assinada com o SK) por meio de TLS junto com o certificado, e o pin deve ser verificado. A verificação não é necessariamente realizada online, ela pode ser realizada posteriormente, ou por um daemon em execução no cliente que baixa informações de verificação de um espelho periodicamente (para manter SKs atualizados para verificação posterior). As mesmas considerações de CT podem ser aplicadas ao SK, uma vez que os dispositivos podem não estar sempre online para manter as informações atualizadas.

O TACK não requer canais laterais, pois o servidor fornece todas as informações necessárias durante o handshake. O servidor distribui a chave de assinatura de domínio conhecida como TSK com um sinalizador de ativação. Os clientes só precisam armazenar e lembrar os TSKs para verificação futura de TACKs assinados com os TSKs armazenados. No entanto, a especificação descreve que os TACKs podem ser baixados opcionalmente de um terceiro confiável. Apesar de não haver uma descrição sobre a forma como os clientes confiam nesses terceiros, fazer o download e manter uma lista de terceiros confiáveis pode ser considerado o uso de um canal lateral. Além disso, o TACK também propõe que os clientes compartilhem seus TACKs observados, exigindo canais laterais adicionais.

CAA é um mecanismo para proprietários de domínio anunciarem qual CA está atualmente autorizada a emitir certificados para seus domínios. O CAA não envolve clientes, portanto, nenhum canal lateral pode ser considerado.

HSTS e HPKP são limitados a HTTP. O primeiro define políticas para impor o uso de TLS e o outro faz uma promessa sobre a cadeia de certificados que um servidor usará no futuro. Como as informações de HSTS e HPKP são fornecidas simultaneamente usando HTTP sobre TLS e não precisam de verificação adicional, este protocolo não depende de canais laterais.

Finalmente, o DANE usa DNS com extensões de segurança para verificar o certificado TLS. Por esse motivo, ele usa um canal lateral (DNS) imediatamente antes do handshake TLS.

B. Instant Recovery From Loss of Key – IR (Recuperação Instantânea de Perda de Chave – IR)

Um dos problemas mais preocupantes da PKI é ter uma chave privada comprometida. Este problema na PKI tem associado vários processos como: revogação de certificado, que envolve a CA ter o certificado adicionado à lista de revogação; solicitação de novo certificado; e restabelecimento do serviço com o novo certificado.

Em caso de perda, o CT exige a emissão de um novo certificado a ser notificado aos logs apropriados. O certificado anterior não é revogado explicitamente, pois a adição de um novo certificado para o nome de domínio revoga implicitamente o certificado anterior. Apesar deste processo poder ser acionado imediatamente após a emissão do novo certificado, deve-se considerar que a adição do novo certificado pode ser adiada até MMD.

Após o MMD, o novo certificado pode ser verificado de forma assíncrona por monitores. Permita que “Atraso Máximo de Verificação” (MVD) seja o momento para verificação. O tempo total de recuperação é limitado a um máximo t = MMD + MVD. Além disso, a verificação de um SCT pode ser acionada por um auditor, solicitando informações a um monitor, que eventualmente chegará a um log. O tempo até que o primeiro cliente se conecte a um servidor que fornece o novo SCT e, assim, um auditor verifica, ou um monitor solicita a verificação de um SCT pode ser denominado tempo de “Entrega máxima de SCT” (MSD) e corresponde a t = MMD + MSD. Por esse motivo, o tempo total é t = máx (MMD + MVD, MMD + MSD). Esse tempo deve ser considerado já que os monitores não precisam verificar todos os servidores de log nem procurar todos os domínios, podendo ocorrer um SCT não ser verificado em muito tempo.

Caso contrário, o SK define um mecanismo de revogação explícito que distingue entre o certificado TLS e a chave SK do domínio. Ele fornece um mecanismo para recuperar o SK usando o “incorporar por referência”. Em geral, é possível usar uma nova chave imediatamente após a perda da anterior. Da mesma forma que o CT, há certas considerações de tempo que devem ser levadas em consideração.

No SK, não há definição para o tempo máximo que pode levar a adição de um novo registro. Pode ser adicionado imediatamente ou em lotes, porém o SK avalia a atualidade dos dados recebidos pelo cliente sendo o período das 24 primeiras horas, em que qualquer informação é considerada fresca, uma janela significativa de oportunidade para um ataque. Apesar de as novas chaves poderem ser usadas imediatamente, as chaves anteriores podem ser consideradas válidas durante pelo menos 24 horas. Portanto, seria o suficiente para evitar que os clientes sob ataque atualizassem as informações do SK durante 24 horas. SK propõe um protocolo de cooperação para manter uma lista de novos servidores TimeLine que podem reduzir o tempo que um servidor TimeLine comprometido é considerado novo. No entanto, os dispositivos IoT não podem garantir que podem comprometer recursos para a cooperação.

TACK permite recuperar de uma perda de chave do servidor, mas não imediatamente. Uma TACK pode ser substituída de acordo com um plano por um mecanismo de rolagem de chave, uma vez que a proposta suporta a sobreposição de chave durante um tempo. Se uma TACK for comprometida (o que equivale a uma chave de servidor comprometida), ela pode ser revogada aumentando o campo “min\_generation”. Depois disso, uma nova chave pode ser anunciada, mas pode levar algum tempo até que a mudança para a nova chave seja efetiva.

O protocolo HTTP Public Key Pinning (HPKP) faz uma promessa sobre a cadeia de certificados que será usada pelo servidor durante um tempo estabelecido no campo “max\_age” da diretiva de pining. Portanto, novas chaves não podem ser incorporadas imediatamente, exceto para clientes que se conectam pela primeira vez.

Em relação ao DANE, uma vez que os clientes consultam as informações do DANE do DNS antes de se conectar ao servidor, o tempo mínimo para incorporar uma nova chave após a perda da chave do servidor é o DANE RR Time To Live (TTL).

C. Global Attack Detection – GA (Detecção de ataque global - GA)

Um ataque global, neste contexto, é um ataque em que o certificado do servidor foi substituído por um malicioso, emitido por uma CA comprometida, e cada cliente que acessa o servidor vítima pode observar o certificado malicioso. Um certificado malicioso, neste caso, passa na verificação de PKI. Por isso, esta seção avalia não só a capacidade das diferentes propostas em defender clientes e proprietários de domínio contra este tipo de ataques, mas também o efeito das propostas sobre domínios que não adotam cada solução.

Devem ser considerados dois casos diferentes. No primeiro, o invasor compromete a CA e tenta se passar por um servidor TLS usando um servidor TLS com o certificado malicioso, mas não adiciona o certificado malicioso à estrutura verificável (se houver) descrita pela proposta. Este ataque será denominado “ataque apenas de personificação”. No segundo, o invasor não apenas compromete a CA e tenta se passar por um servidor TLS usando um certificado malicioso, mas também adiciona o certificado malicioso à estrutura verificável. Este ataque será denominado “ataque de personificação com envenenamento”.

No caso de um ataque apenas de personificação, os clientes de CT podem se defender contra o ataque se e somente se o CT for adotado globalmente pelos proprietários de domínio, uma vez que os clientes rejeitarão qualquer handshake de TLS que não forneça um SCT como prova. Os proprietários de domínio não perceberão o ataque, pois os auditores não notificam os monitores sobre os servidores que não estão em conformidade com o CT ou que estão em conformidade no passado e pararam de fazê-lo. Os clientes que não oferecem suporte ao CT não perceberão o ataque e podem ficar expostos ao ataque por um longo tempo, pois o proprietário do domínio compatível também não pode perceber o ataque, portanto, não pode reagir a ele.

Se o CT for adotado globalmente por clientes, mas opcionalmente por servidores, um servidor TLS legítimo usando um certificado legítimo, mas não disposto a oferecer suporte a CT, será indistinguível de um malicioso. Os clientes não concluirão o handshake, pois não há possibilidade de verificar o certificado por meio de CT, portanto, os proprietários de domínio que não suportam CT serão excluídos do tráfego seguro. Os proprietários de domínio compatíveis com CT também não perceberão o ataque.

Se o CT for adotado globalmente por clientes e servidores, os clientes irão detectar o ataque, mas não os proprietários de domínio, uma vez que não há nenhum vestígio de certificado malicioso nos serviços de registro.

No caso de um ataque de personificação com envenenamento, as considerações anteriores podem ser aplicadas, exceto que os proprietários de domínio têm uma chance de detectar o ataque se monitorarem ativamente cada servidor de log.

O protocolo de fofoca de CT e a interação auditor-monitor são orquestrados em torno da verificação de SCTs, mas não há mecanismo para sinalizar a falta de um SCT ou detectar servidores que entregaram SCTs no passado e pararam de fazê-lo. Se os auditores pudessem notificar os monitores sobre os servidores que param de entregar SCTs, os monitores poderiam investigar o problema e notificar os proprietários de domínio, contribuindo para uma detecção precoce. Infelizmente, essa funcionalidade não é considerada no CT.

A adoção global de TC por IoT / M2M não é plausível devido à heterogeneidade de dispositivos e soluções, portanto, a detecção de ataque global pode não ser viável.

No caso de um ataque apenas de representação, as entidades que apoiam o SK podem verificar se a chave SK associada está em vigor, solicitando todos os registros disponíveis para aquele domínio desde um determinado tempo ou número de sequência. Consequentemente, se o SK for adotado globalmente pelos proprietários de domínio, mas opcionalmente pelos clientes, os clientes com conhecimento do SK detectarão o ataque, pois o certificado TLS fornecido pelo servidor não possui a assinatura do SK.

Se o SK for adotado globalmente pelos clientes, mas não pelos servidores, o SK teria o mesmo problema que o CT, seria suficiente para um invasor personificar o servidor TLS com o certificado malicioso sem oferecer o SK, então os servidores maliciosos não serão distinguidos dos servidores legítimos não usando SK. Se o SK for adotado globalmente por proprietários de domínio e clientes, o ataque será detectado. Em qualquer caso, os proprietários de domínio não perceberão o ataque, pois nenhum novo registro será adicionado ao servidor TimeLine.

Uma representação com ataque de envenenamento no SK é mais complexa de se conseguir, uma vez que se espera que o SK esteja mais protegido e menos exposto do que as chaves do servidor TLS. No entanto, se o SK estiver comprometido e o proprietário do domínio não perceber, o SK não oferece proteção, pois o proprietário do domínio não pode revogar o SK.

A cooperação entre os dispositivos no SK protege contra adulteração do servidor TimeLine, mas não contra um ataque a um servidor TimeLine concreto.

Se o TACK for adotado globalmente pelos clientes, no caso de um ataque apenas de representação, os clientes serão capazes de detectar servidores TLS mudando de um certificado legítimo para um malicioso, já que o último não terá assinatura TSK. Como acontece no CT e no SK, os servidores legítimos que não suportam o TACK serão tratados da mesma forma que os mal-intencionados. Se o TACK se tornar obrigatório para os servidores e opcional para os usuários, apenas os clientes que reconhecem o TACK estarão protegidos desse tipo de ataque. Mesmo se o TACK for adotado por proprietários de domínio e clientes, não há como os proprietários de domínio perceberem que seus servidores TLS estão sendo personificados, a menos que os clientes compartilhem não apenas TACKs observados, mas relatem a ausência de TACKs para um determinado nome de domínio.

Em princípio, a representação com ataque de envenenamento não tem sentido no TACK, uma vez que não existe uma estrutura verificável, mas na prática, poderia ser possível obter o mesmo resultado envenenando terceiros que os clientes podem usar para obter TACKs confiáveis. Não há uma definição clara dos requisitos que uma entidade deve cumprir para se tornar uma parte confiável, nem definição da segurança do protocolo usado para buscar TACKs deles. Em qualquer caso, se o TSK também estiver comprometido, não há como detectar os dois ataques.

O HPKP permite que clientes compatíveis detectem um ataque global se o ataque acontecer depois de eles terem visitado o servidor TLS pela primeira vez, quando obtiverem a cadeia de confiança que será usada no futuro. Os clientes que não reconhecem o HPKP e aqueles que estão visitando o site pela primeira vez após o início do ataque não notariam.

No caso de um ataque apenas de representação, se o proprietário do domínio apoiar o DANE, apenas os clientes que suportam o DANE perceberão o ataque. Os clientes não precisam confiar nas informações armazenadas anteriormente para detectá-las. No entanto, se o proprietário do domínio não oferecer suporte ao DANE, mesmo os clientes DANE não perceberão o ataque. O ataque não pode ser, em princípio, detectado pelo proprietário do domínio, a menos que use o DANE e outra parte que apóie o DANE encontre uma inconsistência entre as informações autoritativas autenticadas fornecidas pelo DNS e as informações obtidas durante o handshake TLS.

Se os clientes adotarem globalmente o DANE, mas não os proprietários de domínio, um cliente DANE não pode estar ciente de um ataque se o proprietário do domínio não oferecer suporte ao DANE. No entanto, o ataque só é eficaz para um proprietário de domínio ciente do DANE se o invasor comprometer não apenas a CA, mas também o DNS do proprietário do domínio (a fim de remover ou alterar as informações do DANE do servidor DNS). O último pode ser considerado um ataque de personificação com envenenamento, que será útil se, e somente se, o invasor não apenas comprometer o servidor DNS, mas também obter acesso aos ZSKs DNSSEC.

D. Targeted Attack – TA (Ataque direcionado - TA)

Um ataque direcionado é semelhante a um ataque global. Nesse caso, o certificado malicioso não pode ser observado por todos os clientes que acessam o servidor vítima, mas por um grupo de dispositivos que constituem o grupo-alvo.

A eficácia de CT, SK, TACK e HPKP, neste caso, é equivalente à do ataque global discutido na Seção V-C. Novamente, como discutido antes, muitas propostas como CT, SK e TACK, que não relatam servidores TLS que deixam de ser compatíveis, terão uma proteção equivalente. Outros, que esperam um comportamento diferente, como o HPKP se visitado pela primeira vez antes do ataque, detectariam esse tipo de ataque prontamente. Qualquer benefício esperado ou benefício potencial da cooperação entre entidades será reduzido neste caso, uma vez que o grupo capaz de observar o ataque é reduzido. Considerando que os dispositivos IoT / M2M, conforme argumentado anteriormente, podem não ser capazes de comprometer recursos para cooperação, o grau de proteção seria o mesmo.

No caso do DANE, uma vez que não há cooperação esperada e os dispositivos não armazenariam nenhuma informação anterior, a proteção é formalmente igual ao ataque global.

E. Trusted Third Party Dependency – TTP (E. Dependência de terceiros confiáveis - TTP)

Esta seção avalia o impacto de terceiros confiáveis adicionais sobre os participantes. Como já foi discutido na Seção III, o número crescente de CAs (TTPs) direta ou indiretamente confiáveis, distribuídos com o software atual, constitui um problema de confiança e é um dos pontos mais fracos da PKI aplicada ao TLS. Soluções de pinagem de certificado foram propostas para superar essa fraqueza da PKI, evitando alguns ataques de forma mais ou menos eficiente. No entanto, algumas das soluções analisadas contam com TTPs adicionais, que podem eventualmente crescer e se tornar um problema. Além disso, o gerenciamento de uma estrutura TTP totalmente nova pode exigir um esforço extra tanto do lado do cliente quanto do servidor.

Os servidores de log em CT usam um par de chaves públicas para assinar o SCT que emitem após a adição de novos certificados TLS. O TC não esclarece como esses novos pares de chaves são obtidos, distribuídos ou revogados, mas pode-se supor que a lista de pares de chaves autorizadas atuais do TC será distribuída junto com o software, da mesma forma que a PKI. Dessa forma, conforme descrito na especificação, o CT exige que os auditores integrados aos clientes tenham conhecimento dos CT TTPs ou que deleguem aos componentes externos, mas não esclarece como a confiança será estabelecida entre os clientes e os auditores externos. Gerenciar listas de CAs é um problema por si só, mas pior em IoT / M2M, pois muitos dispositivos podem obter atualizações mais tarde do que o esperado ou nunca serão atualizados. Por esse motivo, exigir que os dispositivos IoT / M2M sejam atualizados com uma lista TTP adicional constitui um problema adicional.

No caso do SK, cada servidor TimeLine tem seu próprio par de chaves. SK afirma claramente que a lista de servidores TimeLine e suas chaves correspondentes devem ser distribuídas com o software da mesma forma que os certificados confiáveis em PKI. Além disso, os espelhos no SK devem se identificar por meio de seu IP / porta e chave pública, de modo que o SK introduz dois conjuntos de chaves adicionais (servidor TimeLine e espelhos). O gerenciamento e a distribuição das duas listas podem ser um problema para dispositivos IoT / M2M que podem estar frequentemente desatualizados.

O TACK não precisa de TTPs adicionais. Cada domínio tem seu próprio TSK que é distribuído durante a ativação. O domínio é responsável pela distribuição, não exigindo cooperação dos fabricantes de software. Assim como o TACK, o HPKP não apresenta novos TTPs, apenas transmite quais CAs devem ser encontrados na cadeia de certificados fornecida pelo servidor TLS durante um handshake. No entanto, a TACK propõe que terceiros forneçam conjuntos de TACKs confiáveis. Apesar desses terceiros atuarem como meras lojas online para TACKs, a lista de TACKs autorizados pode precisar de algum gerenciamento

O DANE depende do DNSSEC para autenticar as respostas do DANE DNS. Essas respostas, que podem ser usadas para verificar os certificados TLS, são assinadas com um ZSK, que é autorizado pelo pai com uma assinatura, e sucessivamente até a raiz que é assinada por uma CA de renome. A CA que assina a raiz DNS exclusiva é bem conhecida e verificável com a lista de certificados PKI atual. Por esse motivo, o DANE não exige o uso de terceiros de confiança adicionais.

F. Instant Start Up – IS (Inicialização Instantânea - IS)

Esta seção discute se as soluções propostas permitem que um servidor TLS comece a usar um certificado imediatamente após a emissão, sem gerar problemas de confiança nos clientes.

O CT exige que o CA registre o certificado em um servidor de log para obter o SCT, que será entregue por meio de uma extensão em TLS ou diretamente embutido no certificado. Os servidores de log CT adicionam novos certificados de forma assíncrona em um tempo menor que o MMD. Um certificado fornecido por um servidor TLS é confiável se o SCT for assinado por uma chave pública de servidor de log na qual o cliente confia. Se o servidor de log foi comprometido, os participantes podem precisar de até um tempo t = max (MMD + MVD, MMD + MSD), conforme discutido na Seção V-B, para verificar um SCT confiável. Em IoT / M2M, o CT não supõe nenhuma melhoria em relação ao PKI, pois em ambos os casos um certificado TLS pode ser usado imediatamente e o CT oferece o mesmo grau de segurança que o PKI durante t.

SK não impede o uso instantâneo de certificados TLS, uma vez que o único requisito é o proprietário do domínio registrar uma chave de domínio (SK) de antemão. Depois que um proprietário de domínio registra o SK para seu domínio, qualquer novo certificado pode ser usado instantaneamente. O único requisito para uma verificação cruzada é ter o novo certificado assinado com o SK de domínio. Da mesma forma que o CT, se um servidor TimeLine foi comprometido, seria necessário aguardar mais de 24 horas, de acordo com o frescor definido pelo protocolo, para haver uma verificação cruzada confiável. Portanto, o grau de proteção, pelo menos durante uma janela de 24 horas, seria o mesmo que com PKI simples.

No caso do TACK, há um momento em que os TACKs a serem retirados são distribuídos junto com suas reposições, portanto, uma mudança deve ser planejada com antecedência. Assim, é possível começar a usar um novo certificado com seu TACK correspondente imediatamente, mas o processo de distribuição pode afetar o tempo de convergência. Deve-se considerar também que o TACK permite que terceiros forneçam conjuntos TACK confiáveis, portanto, eles devem ser atualizados de acordo com as alterações.

HPKP permite o uso imediato de novos certificados por servidores TLS. A principal preocupação é que os clientes, cuja primeira interação com o servidor ocorre após a mudança, não podem determinar se o certificado é legítimo ou não.

Finalmente, o DANE é considerado um canal independente alternativo. Uma vez que os clientes DANE não armazenarão nenhum registro anterior de um determinado servidor além do tempo TTL do registro, eles tomarão como válido o novo certificado.

G. Unmodified Servers – US (Servidores não modificados - EUA)

Uma das maiores preocupações do Certificate Pinning é a adição ou alteração de protocolos para apoiar as propostas. Mudanças significativas irão adicionar complexidade aos clientes, complicando a adoção. Esta seção analisa as alterações exigidas pelas soluções de pinagem de certificados discutidas anteriormente.

O CT propõe mecanismos alternativos para entregar SCTs ao cliente durante um handshake de TLS. O primeiro consiste em entregar SCTs usando extensões TLS com duas possibilidades, usando uma extensão CT TLS ou uma OSCP. A segunda descreve como o SCT pode ser incorporado diretamente ao certificado, registrando um pré-certificado antes de emitir o certificado final. No primeiro caso, o CT exige que os clientes suportem as extensões TLS descritas na proposta. Apesar de suportar uma extensão TLS exótica pode ser um desafio, o CT considera o uso da extensão TLS OSCP amplamente adotada. Assim, o uso de extensões TLS não é uma grande preocupação, pois são amplamente suportadas. No segundo caso, não há necessidade de alterar o protocolo, uma vez que o SCT está embutido no certificado, então a complexidade é movida para o software PKI.

No entanto, deve-se considerar que a TC propõe um protocolo de fofoca para os clientes trocarem SCTs. Esse novo protocolo deve ser suportado pelos clientes. Em relação ao IoT / M2M, é coerente assumir que o suporte à extensão TLS não é um problema, uma vez que eles são suportados por implementações TLS de referência como OpenSSL. Em relação ao protocolo de fofoca e qualquer outro protocolo de suporte à interação entre as partes (ou seja, auditor-monitor), seu suporte pode ser mais desafiador, uma vez que o software adicional precisa ser atualizado, mas mais precisamente devido às razões discutidas na Seção V-A.

Na especificação SK, sugere-se que a assinatura SK na cadeia de certificados seja transmitida durante o handshake como parte do certificado do servidor. Neste caso, não há necessidade de modificação do protocolo.

Da mesma forma que o CT, o TACK requer o uso de extensões TLS que são bem suportadas na IoT. No entanto, o TACK pode precisar de suporte para terceiros confiáveis, fornecendo TACKs confiáveis aos clientes.

O HPKP e o HSTS exigem mudanças nos cabeçalhos HTTP em clientes e servidores, portanto, os servidores podem entregar a cadeia e os clientes a processam. Os clientes que não conseguem entender esses cabeçalhos devem ignorá-los, para que não seja um problema no IoT / M2M.

Finalmente, o DANE não requer nenhuma modificação no servidor ou clientes. Devido a isso, ele pode ser usado sempre que o resolvedor DNS do cliente suportar DNSSEC e for capaz de processar RRs DANE.

H. Necessary Storage – NS (Armazenamento necessário - NS)

As soluções de pinagem de certificado buscam fornecer algumas das seguintes características: estruturas verificáveis que podem ser auditadas; verificação cruzada para clientes; ou fundamentos para confiança com base em coleções de evidências. Assim, algumas informações devem ser armazenadas em diferentes entidades consideradas pela solução. É necessário avaliar o impacto dessas propostas no armazenamento.

O CT requer que o armazenamento de SCTs no lado do cliente (auditor) seja verificado por monitores ou compartilhado por meio do protocolo de gossip. Além disso, o CT requer o armazenamento de uma lista de TTPs ou servidores de log que devem ser confiáveis. De acordo com o tamanho esperado de SCTs e chaves públicas, não seria um evento problemático para IoT / M2M, a menos que o número de SCTs ou chaves aumentasse exageradamente. Além disso, o CT exige que os auditores armazenem evidências (partes do MT ou sequências de nós) que serão usadas para detectar erros e comportamentos maliciosos.

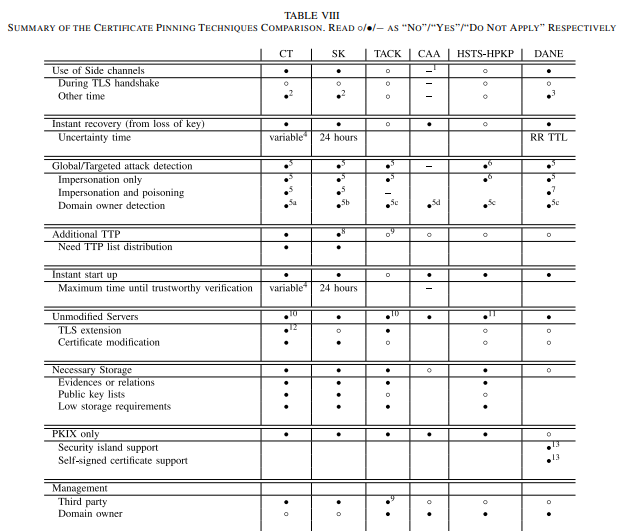
SK requer o armazenamento de TimeStamps, SKs e listas de espelhos com suas chaves correspondentes para alimentar o protocolo de sincronização e permitir a verificação. A especificação faz um estudo do tamanho exigido estimado de acordo com a entropia dos dados, mostrando os requisitos que podem ser atendidos pelos dispositivos IoT / M2M.

A especificação TACK define “armazéns” como o local dentro do cliente onde armazenar TACKs observados e baixados.

De acordo com a estrutura TACK, não deve ser um problema armazenar um número razoável de TACKs em IoT / M2M.

Da mesma forma que as soluções mencionadas acima, o HPKP requer o armazenamento de associações HPKP e políticas HSTS no cliente, mas não deve ser um volume de informações significativo para IoT / M2M.

O DANE não exige que o cliente armazene nenhuma informação.



**VII. CONCLUSÃO**

O TLS tem sido amplamente utilizado na Internet com alto grau de sucesso. Portanto, é razoável que tenha sido adotado pelos protocolos IoT e M2M. No entanto, TLS e PKI lidam com serviços estáticos, enquanto os serviços em IoT / M2M podem ser dinâmicos e várias ordens de magnitude mais numerosos. Além disso, a IoT / M2M requer suporte de segurança adequado para lidar com as limitações de processador, memória e bateria e vários fornecedores.

O quadro global de PKI apresentado no artigo identifica problemas (Seção III-A) no que diz respeito ao número de autoridades que, combinado com o número crescente de endpoints a proteger, revela que mecanismos alternativos de validação de certificado e detecção de ataques são desejáveis para garantir um grau adequado de segurança.

A técnica de pinagem de certificado (Seção IV) está sendo considerada uma forma de fortalecer a confiança no sistema. Este artigo revisou e comparou as diferentes propostas para pinning de certificado no contexto de IoT / M2M. A seção V analisou as propostas em diferentes aspectos. Vamos revisar os principais pontos que encontramos.

Apenas CT, SK e DANE usam canais laterais. CT para trocar dados de SCTs, SK para verificar o pino e DANE para consultas iniciais de DNS. Em caso de perda de chave, CT, SK, CAA e DANE fornecem recuperação instantânea, com tempos de incerteza diferentes, mas limitados.

Em um cenário de ataque global, a adoção global de TC por IoT / M2M é necessária. Isso não é plausível devido à heterogeneidade de dispositivos e soluções, portanto, a detecção de ataques globais pode não ser viável. Problemas semelhantes aparecem para o SK, em que os servidores maliciosos serão indistinguíveis dos servidores legítimos que não usam o SK.

Este também é o caso de TACK se adotado por proprietários de domínio e clientes, pois não há como os proprietários de domínio perceberem que seus servidores TLS estão sendo personificados, a menos que os clientes compartilhem não apenas TACKs observados, mas relatem a ausência de TACKs para um determinado nome de domínio. O HPKP protege apenas os clientes de um ataque global a servidores TLS já visitados e confiáveis. Isso pode limitar fortemente a dinamicidade das interações com a IoT, aumentando o grande número de aplicativos, serviços e dispositivos. O DANE também exige a adoção global em servidores e clientes para proteção contra um ataque apenas de personificação, se o proprietário do domínio apoiar o DANE, apenas os clientes que suportam o DANE perceberão o ataque. Não há diferenças significativas em relação à inicialização instantânea e modificação do servidor.

CT e SK são as únicas abordagens que requerem TTPs adicionais. A necessidade de armazenamento extra é comum a todas as propostas, exceto CAA e DANE, uma vez que obtêm as informações do DNS. O DANE é o único que oferece suporte a ilhas de segurança e cenários de certificados autoassinados de proprietário de domínio. CT, SK e TACK requerem gerenciamento de terceiros, enquanto os outros podem ser gerenciados pelo proprietário do domínio.

De tudo isso, concluímos que o DANE oferece um conjunto de características desejáveis para a IoT e seus cenários dinâmicos. O DANE usa um canal lateral existente (DNS), mas não requer o uso de um novo canal lateral durante o handshake TLS. Uma consulta para o registro de endereço é mais frequentemente realizada antes do handshake TLS. Assim, os serviços podem atualizar certificados com frequência e se recuperar instantaneamente da perda de chaves, sem impor tempos de convergência ou envolver terceiros. O DANE não requer colaboração entre dispositivos limitados de IoT em cenários de ataque global, nem TTPs adicionais, reduzindo os custos de operação e gerenciamento.

DANE: DNS-based Authentication of Named Entities (Autenticação de Entidades Nomeadas baseada em DNS)