

Conceitos envolvendo TOTP como Segundo Fator de Autenticação

Leandro Borges Castilho & Marcelo Vinícius Cysneiros Aragão

Abstract— When searching for TOTP - *Time-Based One-Time Password Algorithm*, commonly used as the second factor of authentication, there are many technical documents with more security-oriented approaches. This document aims to enrich this scenario with a more practical perspective. Taking advantage of the experience gained, working directly with this type of technology, we intend to show the points of the specification that generate more doubts, demonstrating the importance of each concept involved.

Index Terms— Authentication, OTP, RFC, Second Factory Authentication, Token, TOTP.

Resumo— Ao pesquisar sobre a TOTP - *Time-Based One-Time Password Algorithm* - Algoritmo de senha única baseado no tempo, comumente utilizada como segundo fator de autenticação, encontra-se muitos documentos técnicos com uma abordagem mais voltada para segurança. Este documento tem por objetivo enriquecer este cenário com uma perspectiva mais prática. Aproveitando da experiência adquirida, trabalhando diretamente com esse tipo de tecnologia, pretende-se mostrar os pontos da especificação que mais geram dúvidas, demonstrando a importância de cada conceito envolvido.

Palavras Chave— Autenticação, OTP, RFC, Segundo Fator de Autenticação, Token, TOTP.

I. INTRODUÇÃO

A proteção dos dados e sistemas sempre foi uma temática importante, principalmente nos dias de hoje onde a evolução tecnológica e a dinâmica da internet possibilitou o acesso às informações de qualquer lugar. Com tudo isso, a necessidade de proteger e restringir o acesso também precisava evoluir, e a tão usada e conhecida senha já não era mais o suficiente.

A senha, para se tornar mais segura, deveria ser grande, alfanumérica, com caracteres especiais, letras maiúsculas e minúsculas, não ser relacionada com coisas pessoais e mudar com frequência, mas, com mencionado pelos autores Seitz et al. [1], para facilitar a memorização, as pessoas fazem o oposto, o que faz com que seja facilmente descoberta.

Para resolver esse problema veio a autenticação por múltiplos fatores (MFA - *Multi-factor authentication*) que adiciona uma camada extra de proteção ao sistema. Pode ser considerado como fator de autenticação algo que o usuário saiba (a senha), ou algo que ele tem (*token*, telefone, certificado), ou algo que

eles é (biometria). Uma forma de MFA é o segundo fator de autenticação (2FA - *Second-factor authentication*), que combina o uso de dois desses fatores [2].

Com o segundo fator, o *hacker* (aquele que está tentando acessar indevidamente as informações) terá dificuldades, mesmo que ele descubra a senha do usuário, não conseguirá acessar, pois ainda lhe faltaria o segundo fator, que só o usuário tem. O contrário também é válido, uma vez que o *hacker* tenha acesso ao segundo fator, ainda assim faltaria a senha [3].

Este recurso vem sendo oferecido pelos bancos aos clientes [3, 4], assim como há empresas que também fazem o uso do segundo fator para proteger o acesso aos seus sistemas e as redes de dados. Essa ferramenta de segurança está cada dia mais disponível em serviços, como contas de e-mail, redes sociais, contas de jogos e videogames, entre outros [5, 6, 7, 8, 9, 10].

Há vários recursos que podem ser usados com segundo fator. Neste trabalho será apresentado uma tecnologia bastante interessante, que vem desempenhando bem esse papel, a OTP - *One-Time Password Algorithm* - Algoritmo de senha única.

Quando se toma a decisão de usar ou implementar este tipo de solução, faz-se necessário entender determinados conceitos que muitas vezes estão disponíveis em documentos técnicos que na maioria das vezes não tem a pretensão de justificar o porquê de cada recurso.

Este documento apresenta o tema utilizando-se da metodologia da experiência. Ele tem por objetivo de esclarecer os conceitos de forma prática. Os conceitos estão apresentados na forma de casos de uso demonstrando assim os problemas e as soluções existentes, justificando assim a necessidade de cada conceito. Os casos de uso estão em três grandes grupos, tendo quatro casos em cada um e mais três casos adicionais totalizando 15 casos apresentados.

As seções estão apresentadas seguindo uma lógica de disposição baseada em definição, apresentação de caso(s) de uso e considerações. Na seção II. contem a explicação sobre a tecnologia e suas principais características, seguindo pelas seções III., IV., V. e VI. com os conceitos e seus casos de uso, já na seção VII. tem a comparação dos resultados dos três grupos de casos de uso, que o tema é completado na seção VIII. com mais um conceito adicional e seus casos e por fim a seção IX. com a conclusão.

II. OTP - SENHAS DE USO ÚNICO

OTP, algoritmo de senha única. Neste documento, para manter a linguagem do mercado, será cometida uma imprecisão na nomenclatura, pois usará o termo genérico OTP, mas será apresentado uma extensão deste algoritmo, o TOTP.

TOTP, algoritmo de senha única baseado no tempo, foi especificado na RFC6238 [11] e é uma derivação do HOTP, algoritmo de senha única baseado em HMAC que está especificado na RFC4226 [12]. HOTP é um algoritmo OTP baseado em eventos, logo o TOTP utilizou-se deste conceito e incluiu o tempo para reduzir a vida útil desta senha, tornando-o mais seguro, como foi explicado na RFC6238 [11]. RFCs são documentos técnicos desenvolvidos e mantidos pelo IETF [13].

O TOTP, como o próprio nome indica, é uma senha dinâmica, que só pode ser utilizada uma única vez e é baseada no tempo. Este algoritmo foi especificado pelo grupo OATH [14], que foi criado em 2004 para facilitar a colaboração entre fornecedores de tecnologia de autenticação forte [12].

Esses algoritmos são considerados autenticação forte, pois tem um nível maior de segurança envolvido em comparação com a senha comum, considerado autenticação fraca. Pode-se referir como “o” OTP considerando “o código gerado”, mas também pode ser “a” OTP referindo-se como “a senha dinâmica gerada”.

O gerador de OTP, também conhecido como *token*, assim como apresentado na Fig.1, pode ser encontrado na forma de *Device* (chaveiro, o *hardware token*) [15] ou *Mobile* (aplicativos para *smartphones*, o *software token*) [16]. Do outro lado, está o serviço de validação, que verifica se sua OTP é igual a gerada no *token*. A RFC4226 [12] recomenda que a OTP tenha pelo menos 6 dígitos, sendo desejável que o valor seja apenas numérico, de modo que possa ser facilmente inserido em dispositivos restritos, como telefones.



Fig. 1. *Hardware token, Software token.*

Um ponto importante que os autores M'Raihi et al. [12][12] observaram, é que este algoritmo busca apenas a melhoria da segurança do acesso, no que tange a autenticação, ele não substitui as ferramentas de criptografia, para proteção dos dados armazenados nem tão pouco a da comunicação.

A OTP por ser baseada no tempo, tem sua autenticação impactada diretamente por qualquer divergência no horário. Logo serviço e *token* devem compartilhar do mesmo fator de tempo, e como existem divergências com fuso horário, e diferentes representações de horário, a RFC [11] especifica que a linha

base de tempo é a Hora Unix [17], ou seja, o número de segundos decorridos desde a meia-noite UTC de 1 de janeiro de 1970.

Outro ponto muito importante, é que cada *token* tem uma chave única, e a OTP gerada é baseada na combinação da chave e do tempo. Essa combinação dificulta a vida dos *hackers*, pois só o tempo não é suficiente, seria necessário conhecer a chave certa para gerar a mesma OTP. A especificação RFC4226 [12] recomenda que a chave secreta tenha um comprimento de 160 bits (20 bytes).

Como o serviço e o *token* precisam gerar a mesma OTP, logo é necessário que ambos conheçam a mesma chave. Neste ponto a especificação RFC6238 [11] apresenta duas opções. A primeira opção seria a chave aleatória e compartilhada e para isso recomendam o uso de protocolos de comunicação segura para o tráfego dessa chave. A segunda opção seria a geração a partir de um algoritmo de derivação de chaves que ambos conheçam. Para ambas as opções a especificação fala que as chaves devem ser armazenadas em um dispositivo inviolável e devem ser protegidas contra o acesso e uso não autorizado [11].

Do ponto de vista de segurança, pode-se dizer que o *hardware token* é o mais seguro, pelo simples fato de que, para se ter acesso à informação do *token*, seria necessário ter o dispositivo em mãos, para desmontá-lo e então ter acesso a memória [18], para reforçar a segurança a RFC6238 [11] recomenda que o *token* deve ter um sistema que inutiliza os dados em caso de violação. O *software token* assim como o serviço de validação, por sua vez deve fazer uso de todas as tecnologias disponíveis para a proteção dos dados, dentre elas, destacamos primeiramente, a criptografia.

O uso de camadas de criptografia é de extrema importância para salvar os dados armazenados [19]. A chave deve ser descriptografada somente para verificar o valor da OTP e criptografada imediatamente para limitar a exposição na memória RAM a um curto período de tempo, assim como foi dito por M'Raihi et al.[11]. Outro ponto que pode ser utilizado é a ofuscação do código, dificultando quem tentar utilizar o decompilador para entender seu código, pois diminui sua legibilidade [20].

No serviço de validação o *token* será relacionado a um usuário, para que ao digitar o primeiro fator (login e senha) possa ter em mãos a mesma chave para autenticar o segundo fator (OTP digitada). Vale ressaltar que a especificação RFC4226 [12], também recomenda que seja limitado o número de tentativas de autenticação, evitando assim possíveis tentativas dos atacantes por adivinhação. Outra medida de segurança é o bloqueio ou substituição do *token*, para os casos de perda, pois assim evita-se o uso indevido do dispositivo.

Um bom exemplo de uso do segundo fator de autenticação é o acesso a rede de dados apresentado na Fig.2.

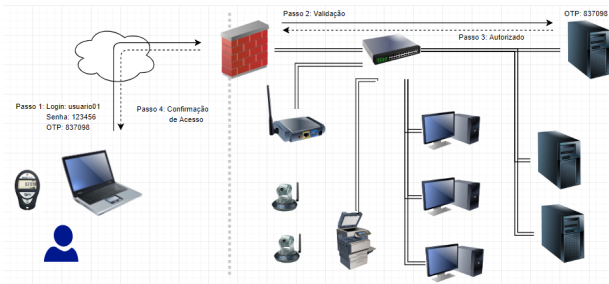


Fig. 2. Exemplo de integração.

O usuário precisa acessar os recursos, mas está fora das dependências da empresa, para isso utiliza a VPN, rede privada virtual, que cria um canal seguro de comunicação para a rede da empresa. O usuário envia seus dados, bem como a OTP gerada em seu *token*, e estas informações são verificadas e a OTP confrontada com a gerada no servidor, estando tudo de acordo o acesso é liberado.

Mesmo com a padronização, pela hora Unix, é uma tarefa bem difícil manter relógios sincronizados perfeitamente, logo é possível haver diferenças de segundos, o que tornaria inviável o processo de autenticação, pois os segundos divergentes gerariam OTPs diferentes.

Para aprofundar o assunto, serão apresentados os casos de uso que utilizarão cada um dos recursos existentes. O primeiro recurso que será usado é o ciclo de vida, exatamente para resolver o problema das pequenas divergências de segundos.

III. CICLO DE VIDA.

Ciclo de vida, também conhecido como *time step*, representado na Fig.3, é a denominação dada ao período de geração de uma OTP, ou seja, o intervalo de tempo em que teremos uma OTP diferente, a RFC6238 [11] também estabelece um valor padrão de 30.

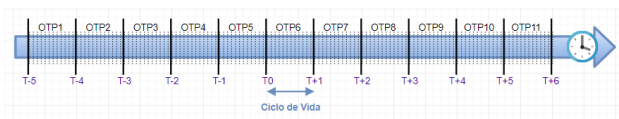


Fig. 3. Ciclo de vida.

Durante o intervalo de tempo entre (T_0) até $(T + 1)$ o *token* e o servidor de validação, irão gerar a *OTP6*, assim como no próximo intervalo, de $(T + 1)$ à $(T + 2)$, a *OTP7* e assim sucessivamente.

Para exemplificar serão montadas três linhas de tempo como apresentado na Fig.4:

- A linha mais a baixo representa o horário do servidor, sendo a base de referência.
- A linha mais a cima representará o horário do *token*.
- E por fim, a linha do meio é uma representação da linha de tempo da autenticação.

As linhas de tempo estarão subdivididas pelo ciclo de vida, que usará o valor padrão (30 segundos), ou seja, variações de 0 a 29 segundos serão toleradas, pois representam a mesma OTP.

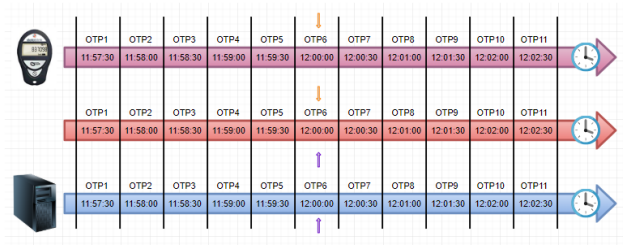


Fig. 4. Linha Base.

Serão apresentadas as ideias em grupos, e a cada grupo um novo conceito será introduzido, para que ao final se possa fazer uma comparação dos resultados apresentados. Os grupos serão apresentados com quatro diferentes ocorrências:

- A primeira ocorrência é o *token* com o relógio interno igual ao relógio do servidor.
- A segunda ocorrência é o *token* com o relógio interno adiantado em relação ao relógio do servidor.
- A terceira ocorrência é o *token* com o relógio interno atrasado em relação ao relógio do servidor.
- A quarta ocorrência é o *token* com o relógio com uma diferença aceitável em relação ao relógio do servidor, mas com um atraso no uso.

O atraso no uso, mencionado anteriormente, pode ser decorrente de uma série de motivos ou combinações entre eles. O usuário pode ter demorado para confirmar a requisição de autenticação, ou pode ter pego a OTP no final do ciclo de vida, ou a requisição de autenticação sofreu com a latência da rede, um atraso na entrega para o servidor.

A. Casos de uso da autenticação pura.

Autenticação pura é a denominação utilizada para descrever o processo de autenticação utilizando apenas o recurso do ciclo de vida.

O objetivo deste grupo é demonstrar os quatro primeiros casos e os problemas que normalmente ocorrem neste cenário de autenticação.

Caso 1. *Token* com o relógio igual.

O caso 1, representado na Fig.5, será apresentado o *token* com o relógio interno igual ao relógio do servidor.

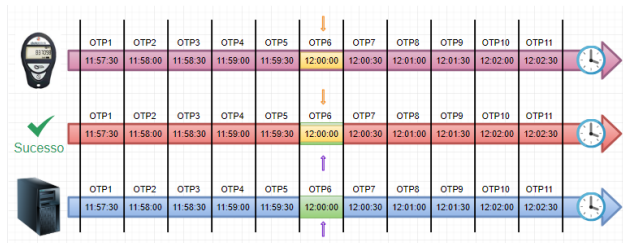


Fig. 5. Caso 1.

É um caso de sucesso na autenticação, pois a OTP gerada no *token* será a mesma gerada no servidor no ato da autenticação ($OTP6 == OTP6$).

Caso 2. *Token* com o relógio adiantado.

O caso 2, representado na Fig.6, apresenta o *token* com o relógio interno adiantado em relação ao relógio do servidor.

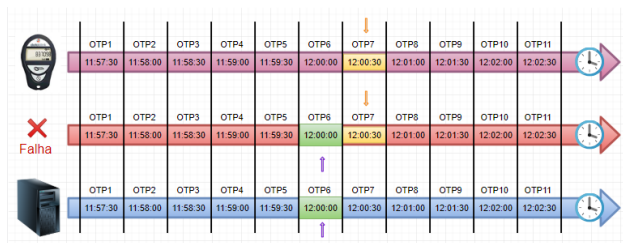


Fig. 6. Caso 2.

É um caso de falha na autenticação, pois a OTP gerada no *token*, não será a mesma gerada no servidor no ato da autenticação ($OTP7 \neq OTP6$).

Caso 3. – *Token* com o relógio atrasado.

O caso 3, representado na Fig.7, é equivalente ao caso 2, apresentando o *token* com o relógio interno atrasado em relação ao relógio do servidor.

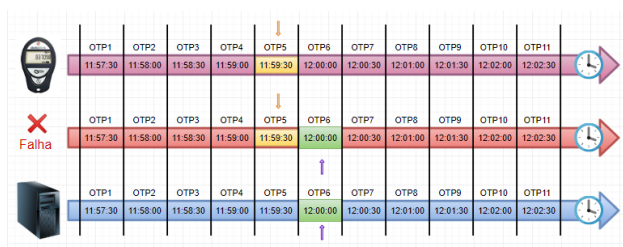


Fig. 7. Caso 3.

Este é mais um caso de falha na autenticação, na mesma ideia do caso 2, pois a OTP gerada no *token* não será a mesma gerada no servidor no ato da autenticação ($OTP5 \neq OTP6$).

Caso 4. – *Token* com o relógio aceitável e atraso no uso.

O caso 4, representado na Fig.8, será apresentado um comportamento diferenciado, pois o *token* está com o relógio interno igual ou com uma diferença aceitável em relação ao relógio do servidor, mas sofrerá com um atraso no uso da OTP.

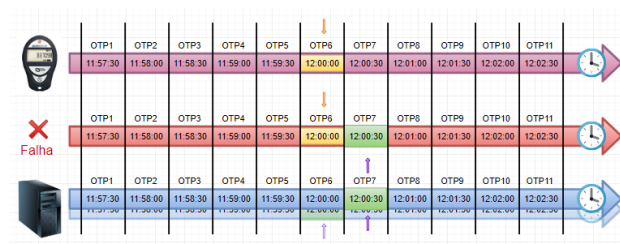


Fig. 8. Caso 4.

O objetivo é demonstrar outro caso de falha na autenticação, que é devido ao tempo que se passou entre a geração da OTP no *token* e seu uso na validação. Como é possível observar, a linha teve seu ponto de referência deslocado, representando a passagem do tempo, logo a OTP gerada no *token* não será mais a mesma gerada no servidor no ato da autenticação ($OTP6 \neq OTP7$).

B. Considerações sobre autenticação pura.

Como foi observado dos quatro casos apresentados, um foi com sucesso e três foram com falha, demonstrando que, uma conferência pura de valores, mesmo utilizando o ciclo de vida, não é suficiente, pois restringe demais a autenticação.

A primeira ideia que vem à cabeça, para resolver os problemas apresentados, é aumentar o ciclo de vida, mas não se pode esquecer que a OTP é de uso único. Para deixar mais clara vamos demonstrar o reuso apresentaremos o problema do ciclo de vida longo.

IV. REÚSO.

Reuso é a denominação dada quando uma OTP é utilizada mais de uma vez no mesmo ciclo de vida, ou seja, mesmo que as condições sejam atendidas, não se pode validar mais de uma vez a mesma OTP, logo o horário da última autenticação deve ser registrado para impedir o reuso [11]. Sendo a OTP de uso único, um ciclo de vida grande, impossibilitaria o usuário de acessar novamente o recurso durante este período.

A. Casos de uso do reuso na autenticação.

Reuso na autenticação é a denominação utilizada para descrever o processo de autenticação utilizando a mesma OTP.

O objetivo é demonstrar o caso adicional demonstrado o problema do reuso na autenticação e para isso será aplicado uma janela de autenticação maior.

Caso 5. *Token* com o relógio igual e ciclo de vida grande.

No caso 5 teremos o *token* com o relógio interno igual ao relógio do servidor, mas com um ciclo de vida grande. Exemplo: 3 minutos.

1ª Autenticação - É apresentando a hipótese de um usuário que acessou o sistema as 11h:58m:15s, logo utilizou a *OTP2*, representado na Fig.9, para se autenticar com sucesso.

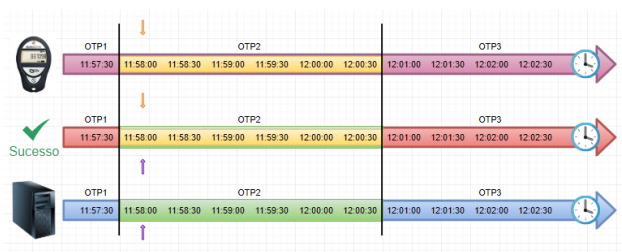


Fig. 9. Caso 5.1.

2ª Autenticação - Imaginando que por qualquer que seja o motivo ele perdeu o acesso, logo precisa se autenticar novamente e neste cenário se a autenticação acontecer durante o período da primeira autenticação.

Como é possível observar na Fig.10, uma autenticação as 12h:00m:25s resultaria em uma *OTP2* já utilizada anteriormente, resultando em falha de autenticação por reuso da OTP.

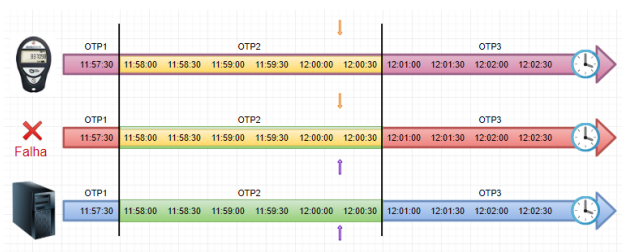


Fig. 10. Caso 5.2.

3ª Autenticação - Sendo assim o usuário só conseguirá autenticar novamente, após as 12h:00m:59s, pois só então terá uma nova OTP.

Como pode ser visto na Fig.11, às 12h:01m:15s temos a *OTP3* e a autenticação ocorre com sucesso.

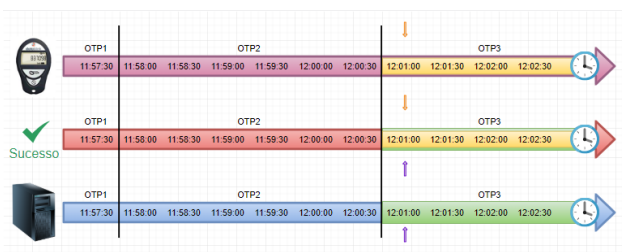


Fig. 11. Caso 5.3.

B. *Considerações sobre reuso de autenticação.*

Como foi observado o reuso, que é um conceito importante que fortalece a segurança, impede que uma janela de autenticação maior seja utilizada. e de acordo com os autores da RFC6238 [11], fazer o usuário esperar não seria a melhor solução.

Para melhorar esta situação faz-se necessário o uso de um outro recurso, a janela de autenticação.

V. JANELA DE AUTENTICAÇÃO.

Janela de autenticação é o nome dado ao intervalo de tempo que as OTPs são válidas, ou seja, é possível autenticar não apenas a OTP daquele momento, mas com um intervalo de possíveis OTPs [11], como está representado na Fig.12.

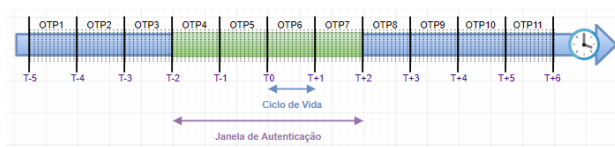


Fig. 12. Janela de Autenticação.

No processo de autenticação as OTPs geradas no intervalo de $(T - 2)$ à $(T + 2)$ são consideradas válidas, logo se o servidor receber a *OTP4*, *OTP5*, *OTP6* ou a *OTP7* o processo ocorrerá com sucesso.

A. *Casos de uso da autenticação com janela.*

Autenticação com janela é a denominação utilizada para descrever o processo de autenticação utilizando o recurso do ciclo de vida e com a janela de autenticação.

O objetivo deste grupo é apresentar mais quatro casos e demonstrar as melhorias quando se utiliza uma janela de autenticação, e os problemas que ainda persistirão.

Vale ressaltar que no grupo anterior de autenticação pura, deu-se a entender que se tratava de uma autenticação sem janela, mas mesmo quando este parâmetro (janela de autenticação) está em zero, ainda assim existe uma tolerância de 30 segundos imposta pelo próprio ciclo de vida da OTP, no entanto, tal tolerância só contempla a OTP corrente.

Os casos de uso serão trabalhados com uma janela de autenticação igual a 150 segundos, sendo +75 segundos, para cobrir os adiantamentos do *token*, e -75 segundos, para cobrir os atrasos do *token* e ou os atrasos de uso em geral. Em termos práticos, esse valor é usado para o cálculo do número de possíveis OTPs, onde a janela de autenticação é dividida pelo ciclo de vida. Para o valor de 150 segundos, dividido por 30 segundos, equivale a dizer que existirá cinco OTPs válidas, sendo uma OTP do momento e duas para os atrasos e duas para os adiantamentos.

Caso 6. Com janela e *token* com o relógio igual.

O caso 6, representado na Fig.13, trabalhará com janela de autenticação e *token* com o relógio interno igual ao relógio do servidor.

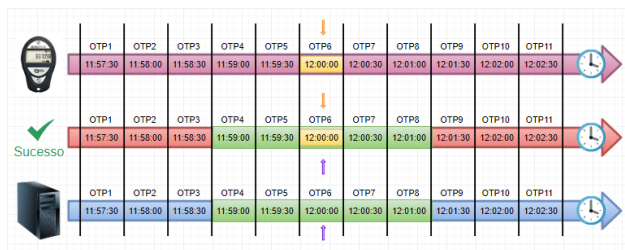


Fig. 13. Caso 6.

A autenticação, assim como no caso 1, ocorre com sucesso. Como pode-se observar, a janela de autenticação possibilitou um intervalo de OTPs. Logo a OTP gerada no *token* é compatível com as geradas no servidor no ato da autenticação ($OTP6 == (OTP4, OTP5, OTP6, OTP7, OTP8)$).

Caso 7. Com janela e *token* com o relógio adiantado.

O caso 7, representado na Fig.14, irá trabalhar com janela de autenticação e *token* com o relógio interno adiantado em relação ao relógio do servidor.

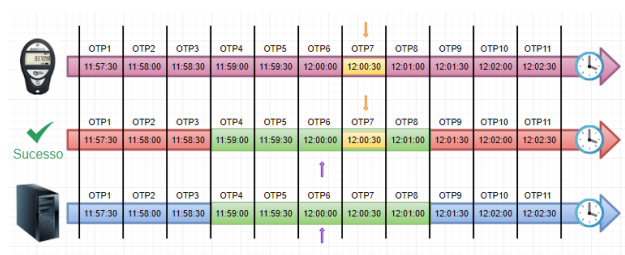


Fig. 14. Caso 7.

A autenticação, ao contrário do caso 2, passou a ocorrer com sucesso. Assim como foi observado anteriormente, há um intervalo de possíveis OTPs, logo a OTP gerada no *token*, mesmo com a divergência no horário, continua compatível com as geradas no servidor no ato da autenticação ($OTP7 == (OTP4, OTP5, OTP6, OTP7, OTP8)$).

Caso 8. Com janela e *token* com o relógio atrasado.

O caso 8, representado na Fig.15, é equivalente ao caso 7 e trabalha com janela de autenticação e *token* com o relógio interno atrasado em relação ao relógio do servidor.

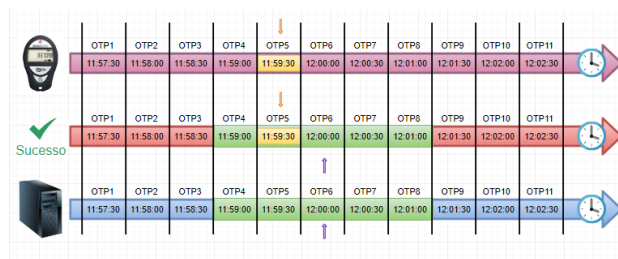


Fig. 15. Caso 8.

A autenticação, assim como no caso 7, passou a ocorrer com sucesso. Assim como foi observado anteriormente, há um intervalo de possíveis OTPs, logo a OTP gerada no *token*, mesmo com a divergência no horário, continua compatível com as geradas no servidor no ato da autenticação ($OTP5 == (OTP4, OTP5, OTP6, OTP7, OTP8)$).

Caso 9. Com janela, *token* com o relógio aceitável e atraso no uso.

O caso 9, representado na Fig.16, tem aquele comportamento diferenciado equivalente ao caso 4, pois trabalha com janela de autenticação, *token* com o relógio interno com uma diferença aceitável em relação ao relógio do servidor e adiciona um atraso no uso da OTP.

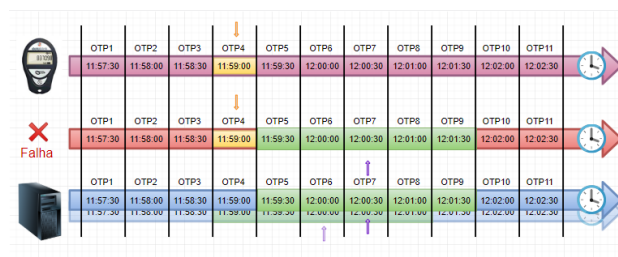


Fig. 16. Caso 9.

A autenticação continua com falha, equivalente ao caso 4. Como pode-se observar, a linha sofreu um deslocamento de seu ponto de referência devido a passagem do tempo, causado pelo atraso do uso, com isso, mesmo tendo um intervalo de OTPs geradas no ato de autenticação, este intervalo já não é mais compatível com a OTP gerada no *token* ($OTP4 == (OTP5, OTP6, OTP7, OTP8, OTP9)$).

B. Considerações sobre autenticação com janela.

Como foi observado nos quatro casos apresentados, três foram com sucesso e um foi com falha, demonstrando uma melhoria significativa no processo autenticação.

Para resolver esta última falha, será feito o uso de um outro recurso, o desvio.

VI. DESVIO.

Desvio, representado na Fig.17, é o nome dado a diferença de horário entre o servidor e o *token* [11].

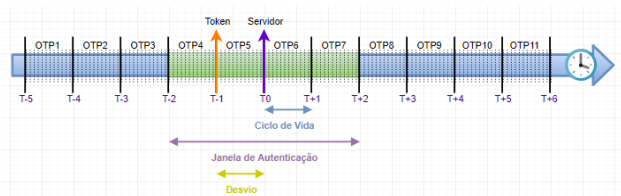


Fig. 17. Desvio.

O desvio é a diferença entre o horário do *token* menos o horário do servidor $(T - 1) - (T_0)$, logo pode ser positivo (relógio do *token* adiantado em relação ao servidor) ou negativo (relógio do *token* atrasado em relação ao servidor). Para ser mais preciso, ele é a diferença de horário ocorrido no ato da autenticação, sendo afetado pelo atraso no uso da OTP. Seu valor é calculado em todas as autenticações realizadas com sucesso, uma vez que o servidor confere apenas o período da janela de autenticação, onde encontram-se as OTPs válidas. Após a autenticação tem-se em mãos o desvio, logo pode ser feito um deslocamento temporal nas próximas autenticações para simular o cenário ideal, onde o *token* tem o mesmo horário do servidor.

A. Casos de uso da autenticação com janela e utilizando o desvio.

Autenticação com janela e utilizando o desvio é a utilização no processo de autenticação dos recursos de ciclo de vida com a janela de autenticação e o desvio.

O objetivo deste grupo é apresentar mais quatro novos casos e demonstrar as melhorias quando se utiliza uma janela de autenticação com desvio. Neste grupo, assim como no grupo anterior, há uma janela de autenticação igual a 150 segundos e é aplicado um deslocamento pelo desvio.

Caso 10. Com janela, utilizando desvio e *token* com o relógio igual.

O caso 10, representado na Fig.18, é igual ao caso 6, com janela de autenticação, utilizando o desvio e *token* com o relógio interno igual ao relógio do servidor.

Aqui o caso está apresentado apenas para manter a coerência, pois o objetivo do grupo é demonstrar o conceito de desvio, este caso não apresenta desvio, uma vez que se faz necessário uma diferença de horário.

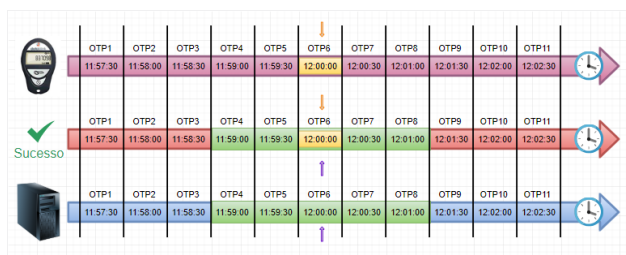


Fig. 18. Caso 10.

A autenticação, assim como no caso 1 e no caso 6, ocorre com sucesso. Como pode-se observar, a janela de autenticação possibilitou um intervalo de OTPs, logo a OTP gerada no *token* é compatível com as geradas no servidor no ato da autenticação ($OTP6 == (OTP4, OTP5, OTP6, OTP7, OTP8)$). Finalizando a autenticação com uma diferença de 0 (zero) segundos.

Como foi explicado anteriormente só a partir de uma primeira autenticação com diferença de horário é que se tem o desvio para o uso nas próximas autenticações, então nos próximos casos serão apresentados mais de uma autenticação.

Caso 11. Com janela, utilizando desvio e *token* com o relógio adiantado.

O caso 11 irá trabalhar com janela de autenticação, utilizando o desvio e *token* com o relógio interno adiantado em relação ao relógio do servidor.

1ª Autenticação - Nesta etapa, representada na Fig.19 é a autenticação que faz o registro do desvio.

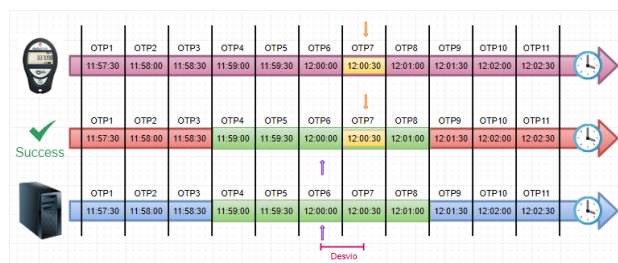


Fig. 19. Caso 11.1.

A autenticação ocorre com sucesso, igual ao caso 7. A OTP gerada no *token*, é compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP7 == (OTP4, OTP5, OTP6, OTP7, OTP8)$). Baseado nesta autenticação chega-se a um desvio de +30 segundos, logo para a próxima autenticação este deslocamento será aplicado.

2ª Autenticação - Nesta etapa já temos o desvio inicial, representado na Fig.20.

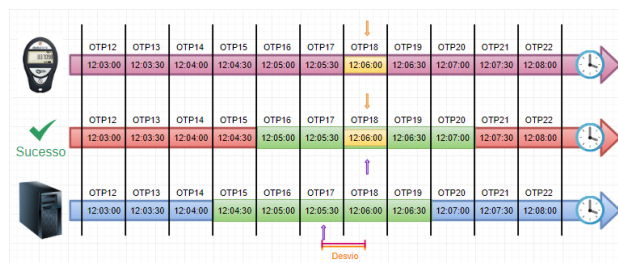


Fig. 20. Caso 11.2.

A autenticação continua ocorrendo com sucesso, e é possível observar que, virtualmente tem-se o mesmo horário, devido ao deslocamento de +30 segundos aplicado pelo desvio, assim a OTP gerada no *token*, continua compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP18 ==$

($OTP_{16}, OTP_{17}, OTP_{18}, OTP_{19}, OTP_{20}$)). Ao final desta validação o desvio permanece com o mesmo valor +30 segundos, pois o deslocamento virtual não entra no cálculo.

Caso 12. Com janela, utilizando desvio e *token* com o relógio atrasado.

O caso 12 é equivalente ao caso 11 e trabalha com janela de autenticação, utilizando o desvio e *token* com o relógio interno atrasado em relação ao relógio do servidor.

1ª Autenticação - Nesta etapa, representada na Fig.21 é a autenticação que faz o registro do desvio.

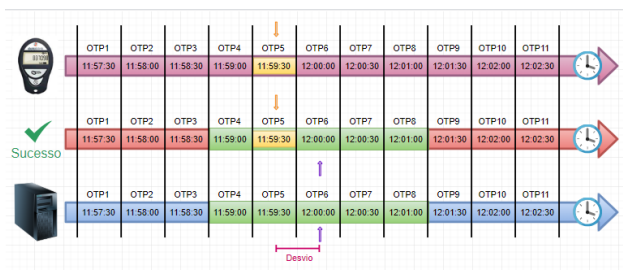


Fig. 21. Caso 12.1.

A autenticação ocorre com sucesso, igual ao caso 8. A OTP gerada no *token*, é compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP_5 == (OTP_4, OTP_5, OTP_6, OTP_7, OTP_8)$). Então chega-se a um desvio de -30 segundos, que será usado na próxima autenticação.

2ª Autenticação - Nesta etapa já temos o desvio inicial, representado na Fig.22.

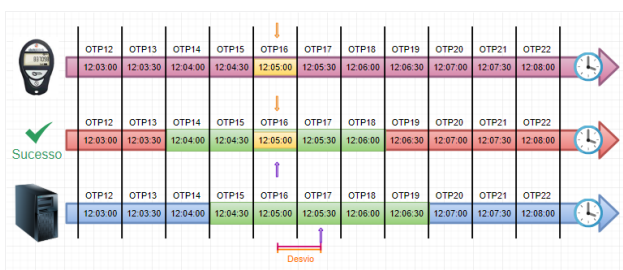


Fig. 22. Caso 12.2.

A autenticação também ocorre com sucesso, e é possível observar que, virtualmente tem-se o mesmo horário, devido ao deslocamento de -30 segundos aplicado pelo desvio, assim a OTP gerada no *token*, também continua compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP_{16} == (OTP_{14}, OTP_{15}, OTP_{16}, OTP_{17}, OTP_{18})$). Permanecendo com os mesmos -30 segundos de desvio após a segunda autenticação. Para ampliar um pouco mais esta ideia, este caso seguirá nesta mesma linha, e adicionará mais uma

autenticação considerando que o *token* sofreu com mais um atraso.

3ª Autenticação - Nesta etapa será aplicado um novo atraso, representado na Fig.23.

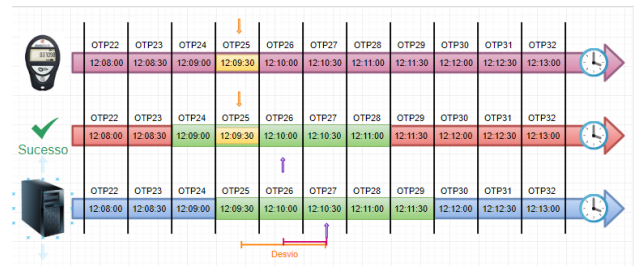


Fig. 23. Caso 12.3.

A autenticação continua ocorrendo com sucesso. Como é possível observar o deslocamento virtual iniciou-se com o desvio de -30 segundos calculado na segunda autenticação, assim a OTP gerada no *token*, mesmo sofrendo mais um atraso continua compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP_{25} == (OTP_{24}, OTP_{25}, OTP_{26}, OTP_{27}, OTP_{28})$). Agora há um novo desvio de -60 segundos, por consequência deste novo atraso, lembrando mais uma vez que o deslocamento virtual não faz parte do cálculo. Assim nas próximas autenticações será utilizado o novo deslocamento.

4ª Autenticação - Nesta etapa já temos o segundo desvio aplicado, representado na Fig.24.

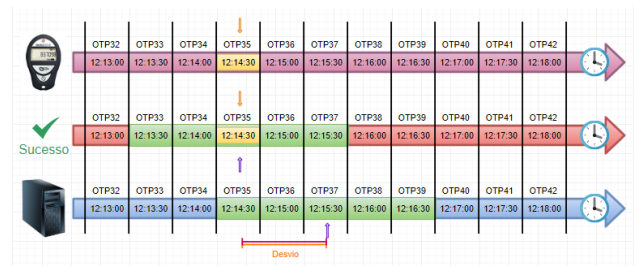


Fig. 24. Caso 12.4.

A autenticação continua ocorrendo com sucesso, como pode ser visto há um reposicionamento da linha virtual de autenticação pelo desvio de -60 segundos devido a terceira autenticação, que simula virtualmente o mesmo horário entre *token* e servidor, assim OTP gerada no *token*, continua compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP_{35} == (OTP_{33}, OTP_{34}, OTP_{35}, OTP_{36}, OTP_{37})$). Ao final desta validação o desvio permanece com os mesmos -60 segundos.

Caso 13. Com janela, utilizando desvio, *token* com o relógio aceitável e atraso no uso.

O caso 13 também traz aquele comportamento diferenciado, com janela de autenticação, utilizando o desvio, *token* com o relógio interno com uma diferença aceitável em relação ao relógio do servidor e com um atraso no uso da OTP.

1ª Autenticação - Nesta etapa será desconsiderado o atraso no uso, pois a intenção é apenas demonstrar o cálculo inicial do desvio, representado na Fig.25.

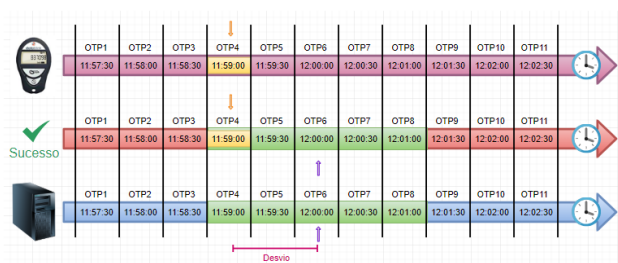


Fig. 25. Caso 13.1.

A autenticação ocorre com sucesso, igual ao caso 8 e caso 12. A OTP gerada no *token*, é compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP4 == (OTP4, OTP5, OTP6, OTP7, OTP8)$). Então chega-se a um desvio de -60 segundos, que será usado na próxima autenticação.

2ª Autenticação - Nesta etapa também será desconsiderado o atraso no uso, pois a intenção é apenas a demonstrar o deslocamento inicial do desvio, representado na Fig.26.

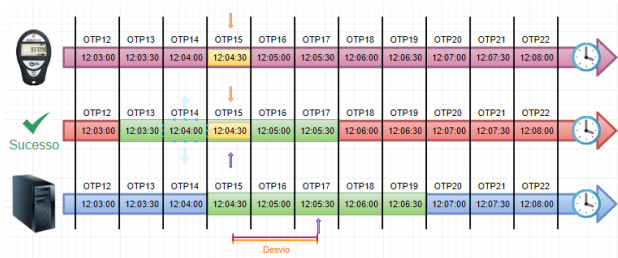


Fig. 26. Caso 13.2.

Igualmente ao caso 12 a autenticação ocorre com sucesso. Como pode ser observado partiu-se com o deslocamento virtual baseado no desvio de -60 segundos calculado na primeira autenticação, assim a OTP gerada no *token*, mesmo sofrendo mais um atraso continua compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP15 == (OTP14, OTP15, OTP16, OTP17, OTP18)$). Agora tem-se um novo desvio de -60 segundos.

3ª Autenticação - Nesta etapa já pronto para demonstrar o atraso no uso, representado na Fig.27.

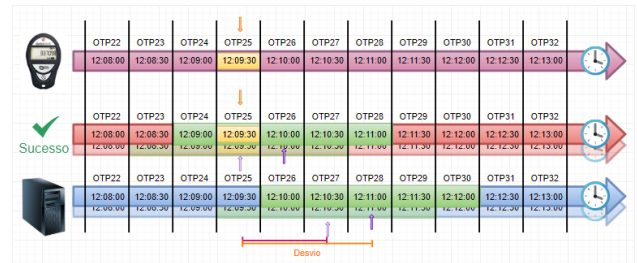


Fig. 27. Caso 13.3.

A autenticação ocorre com sucesso, diferentemente do caso 1 e caso 9. Como pode ser visto, a linha partiu com o deslocamento virtual baseado no desvio de -60 segundos calculado na segunda autenticação e sofreu um deslocamento de seu ponto de referência devido a passagem do tempo devido ao atraso no uso, mas o recurso do desvio permitiu que, mesmo com todas as ocorrências, o intervalo de OTPs geradas no ato de autenticação, permanecesse compatível com a OTP gerada no *token* ($OTP25 == (OTP24, OTP25, OTP26, OTP27, OTP28)$). Finalizando a autenticação com uma diferença de -90 segundos.

4ª Autenticação - Nesta etapa é apenas para demonstrar que o comportamento do desvio se normaliza naturalmente quando o atraso no uso deixa de existir, representado na Fig.28.

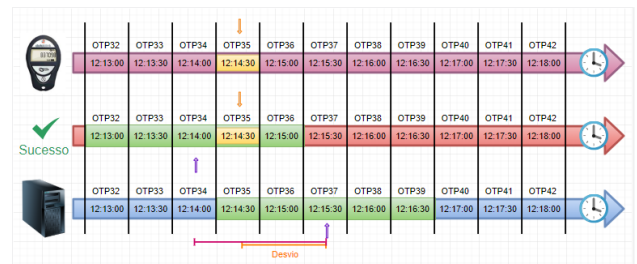


Fig. 28. Caso 13.4.

A autenticação novamente ocorre com sucesso. Como é possível observar o deslocamento virtual partiu do desvio de -90 segundos calculado na terceira autenticação, assim a OTP gerada no *token* é compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP35 == (OTP32, OTP33, OTP34, OTP35, OTP36)$). Agora sem o atraso de uso o desvio volta o desvio aos mesmos -60 segundos que é o atraso real do *token*.

B. Considerações sobre autenticação com janela e utilizando o desvio.

Como podemos observar dos quatro casos apresentados, os quatro foram com sucesso, demonstrando que a combinação dos recursos permite um processo de autenticação capaz de suportar as ocorrências mais comuns.

TABELA I
 COMPARATIVO DOS RESULTADOS

Caso(s) de uso Ocorrência	Autenticação Pura	Autenticação com Janela	Autenticação com Janela e utilizando o desvio
<i>token</i> com o relógio igual.	Sucesso	Sucesso	Sucesso
<i>token</i> com o relógio adiantado.	Falha	Sucesso	Sucesso
<i>token</i> com o relógio atrasado.	Falha	Sucesso	Sucesso
<i>token</i> com o relógio aceitável e atraso no uso.	Falha	Falha	Sucesso

VII. RESULTADOS.

A tabela abaixo é para demonstrar resumidamente todas as melhorias ocorridas pela utilização dos conceitos apresentados.

Há casos em que a divergência no horário pode extrapolar a janela de autenticação, para esses casos a RFC6238 [11] recomenda a criação de medidas adicionais para correção, que no caso será utilizado um outro recurso denominado de janela de sincronismo.

VIII. JANELA DE SINCRONISMO.

A janela de sincronismo, representado na Fig.29, é o intervalo de tempo disponibilizado para que se localize o tempo referente ao relógio interno do *token*, na prática, ele fará uma ampliação temporária na janela de autenticação.

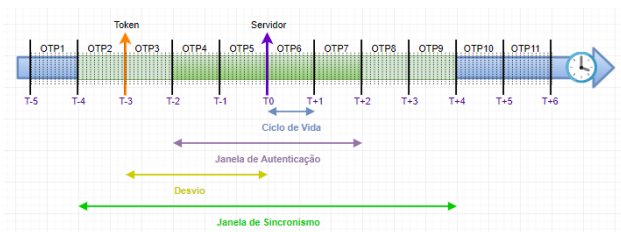


Fig. 29. Janela de Sincronismo.

A janela de sincronismo ao ampliar a janela de autenticação possibilitando que o desvio seja identificado, assim nas próximas autenticações, a janela de autenticação volta aos patamares normais e o desvio passa a exercer seu papel, deslocando a janela para manter virtualmente o mesmo horário.

A. Casos de uso da autenticação com janela de sincronismo.

Autenticação com janela de sincronismo é a utilização no processo de autenticação completo com a abertura temporária da janela de autenticação.

O objetivo deste grupo é apresentar mais dois casos adicionais para demonstrar as a aplicação deste novo conceito.

Caso 14. Com janela, utilizando desvio, utilizando o sincronismo e *token* com o relógio muito atrasado.

O caso 14 será usado para corrigindo o *token* utilizando janela de sincronismo, tendo o *token* com o relógio interno muito atrasado em relação ao relógio do servidor.

1ª Autenticação - Nesta etapa será de demonstrado a falha devido ao grande atraso, representado na Fig.30, para depois aplicar a janela de sincronismo e por fim demonstrar a normalização da autenticação.

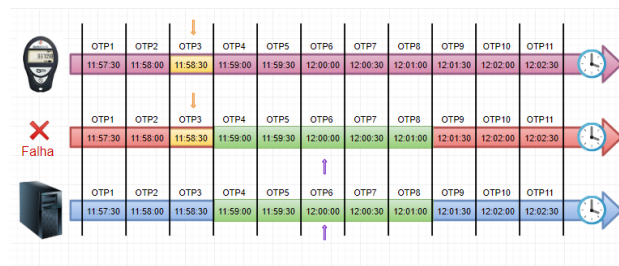


Fig. 30. Caso 14.1.

Neste cenário a autenticação falha, pois a OTP gerada no *token*, não é compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP3 == (OTP4, OTP5, OTP6, OTP7, OTP8)$). Neste processo temos uma diferença de -90 segundos, lembrando, a janela de autenticação contempla apenas atrasos de no máximo -75 segundos, outro ponto é que não houve uma autenticação anterior com sucesso, logo, não foi possível determinar o desvio do *token*, para este caso devemos aplicar a janela de sincronismo.

2ª Autenticação - Para esta etapa será usado uma janela de sincronismo de 270 segundos, representada na Fig.31.

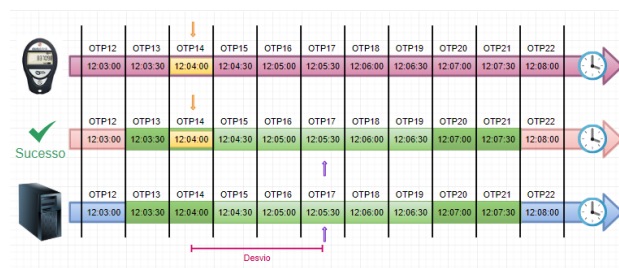


Fig. 31. Caso 14.2.

A autenticação ocorre com sucesso. Como pode ser observado a janela de autenticação temporariamente ampliada permitiu que a OTP gerada no *token*, fosse compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP14 == (OTP13, OTP14, OTP15, OTP16, OTP17, OTP18, OTP19, OTP20, OTP21)$). Finalizando a autenticação com uma diferença de -90 segundos.

3ª Autenticação - Esta etapa será usado apenas para demonstrar o retorno da janela de autenticação ao valor inicial (150 segundos) e a autenticação retornando à normalidade, representado na Fig.32.

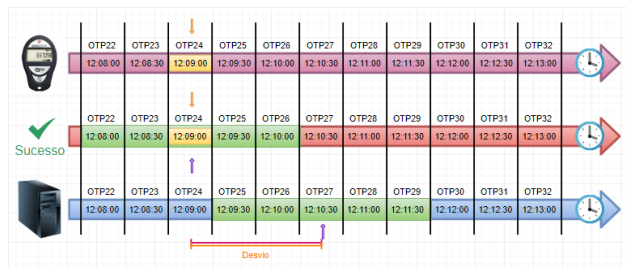


Fig. 32. Caso 14.3.

A autenticação continua ocorrendo com sucesso. Como pode ser visto que a janela de autenticação voltou para 150 segundos e o deslocamento virtual baseado no desvio de -90 segundos calculado na segunda autenticação, possibilitou que a OTP gerada no *token* ficasse compatível com o intervalo de OTPs geradas no servidor no ato da autenticação ($OTP_{24} == (OTP_{22}, OTP_{23}, OTP_{24}, OTP_{25}, OTP_{26})$). Finalizando a autenticação com uma diferença de -90 segundos.

Um recurso muito bom para ser utilizado na validação após o uso da janela de sincronismo, é a autenticação por duas OTPs.

Caso 15. Com janela, utilizando desvio, utilizando o sincronismo por duas OTPs e *token* com o relógio muito atrasado.

O caso 15 é equivalente ao caso 14, também usado para corrigindo o *token* utilizando janela de sincronismo, tendo o *token* com o relógio interno muito atrasado em relação ao relógio do servidor, mas utilizará duas OTPs consecutivas para validar, representado na Fig.33.

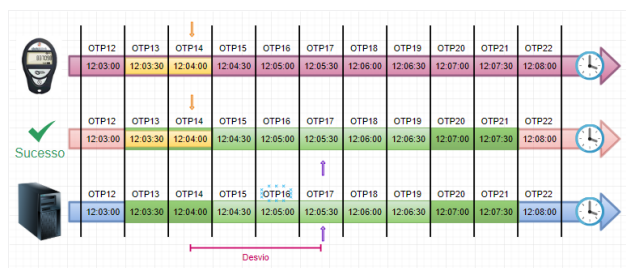


Fig. 33. Caso 15.

A autenticação por duas OTPs eleva o nível de segurança, pois exige que o usuário pegue a primeira OTP gerada e aguarde, o ciclo de vida, para que a próxima OTP seja gerada, para só então utilizar a duas no processo de autenticação, assim eliminando a possibilidade de ser uma OTP gerada ao acaso ou uma antiga guardada. O serviço após encontrar o horário referente a primeira OTP, valida a segunda OTP considerando mais um ciclo de vida de avanço.

B. Considerações sobre autenticação com janela de sincronismo.

Os dois últimos casos mostraram como resolver os grandes desvios, mas fica a pergunta. ‘Em que situação isso acontece?’, pois, os atrasos consecutivos são resolvidos pelo desvio que desloca consecutivamente a janela virtual de autenticação. As grandes diferenças podem ser observadas em situações esporádicas, por conta de mudanças manuais no relógio do *smartphone*. A principal causa dessa mudança deve-se às alterações equivocadas por motivo de viagem do usuário para locais de fuso horário diferente. Outro motivo dessa mudança manual é por conta do horário de verão (o correto é alterar o GMT do relógio e não a hora). O ponto mais importante onde as grandes diferenças são observadas é no primeiro uso, pois não se conhece o relógio interno do *token*. Então é recomendado iniciar na primeira autenticação do *token* com uso da janela de sincronismo, para evitar que ocorra falha e necessite a abertura de um chamado para executar este procedimento.

IX. CONCLUSÃO

O algoritmo OTP, precisamente o TOTP, a senha dinâmica de uso único baseado no tempo, que vem sendo utilizada como segundo fator de autenticação, mostrou-se adequado do ponto de vista de segurança.

O TOTP tem como base de seu segredo, numa composição de dois valores, a chave secreta e o tempo. A chave secreta, sendo um número grande, torna difícil a sua descoberta, e o tempo, por sua vez, faz com que eleve o nível de segurança, tornando menor o ciclo de vida da OTP.

Mesmo sendo bem complexa a utilização do fator tempo, foram criados mecanismos que permitiram saídas elegantes e correções automáticas de eventuais divergências no tempo.

Outro ponto que torna este algoritmo importante como recurso de segurança é o fato de o código gerado não poder ser reutilizado.

Ficou claro também, que este é um recurso somente para autenticação e que não substitui as demais medidas de segurança, inclusive são recomendadas medidas adicionais para proteção das chaves, como criptografia e canais seguros.

Com os dados apresentados, é possível compreender seu funcionamento e assim ter um embasamento para a tomada de decisão na hora de escolher uma tecnologia para usar como segundo fator de autenticação, bem como deixar em alerta sobre as demais medidas necessárias de proteção, tanto para implantar como para verificar em seus fornecedores.

GLOSSÁRIO

REFERÊNCIAS

2FA *Second-factor authentication* - Segundo fator de autenticação. 1

ERP *Enterprise Resource Planning* - Sistema Integrado de Gestão Empresarial. 13

FAI Centro de Ensino Superior em Gestão, Tecnologia e Educação. 13

HMAC *Hash-based Message Authentication Code* - Código de autenticação de mensagens baseado em função hash criptográfico. 2

HOTP *HMAC-Based One-Time Password Algorithm* - Algoritmo de senha única baseado em HMAC. 2

IETF *Internet Engineering Task Force* - Força-tarefa de engenharia da internet. 2

Inatel Instituto Nacional de Telecomunicações. 13

MFA *Multi-factor authentication* - Autenticação por múltiplos fatores. 1

OATH *Initiative for Open Authentication* - Iniciativa para autenticação aberta. 2

OTP *One-Time Password Algorithm* - Algoritmo de senha única. 1–11

PMI *Project Management Institute* - Instituto de Gerenciamento de Projetos. 13

RADIUS *Remote Authentication Dial In User Service* - Protocolo de rede que fornece gerenciamento centralizado de autenticação. 13

RFC *Request for Comments* - Documentos técnicos. 1, 2

SDK *Software development kit* - Kit de desenvolvimento de software. 13

TOTP *Time-Based One-Time Password Algorithm* - Algoritmo de senha única baseado no tempo label. 1, 2, 11, 13

UTC *Coordinated Universal Time* - Tempo universal coordenado. 2

VPN *Virtual Private Network* - Rede privada virtual. 3

Webservice solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. 13

- [1] Tobias Seitz, Florian Mathis, and Heinrich Hussmann. The bird is the word: A usability evaluation of emojis inside text passwords. dezembro 2017. doi: <https://doi.org/10.1145/3152771.3152773>. ISBN: 978-1-4503-5379-3.
- [2] Seth Rosenblatt and Jason Cipriani. Two-factor authentication: What you need to know (faq). *CNET*, 15 de junho 2015. URL <https://www.cnet.com/news/two-factor-authentication-what-you-need-to-know/>. Acesso em: 03 de fev.2019.
- [3] Margaret Rouse. multifactor authentication (mfa). *Techtarget*, março 2015. URL <https://searchsecurity.techtarget.com/definition/multifactor-authentication-MFA>. Acesso em: 03 de fev.2019.
- [4] Token físico (chave de segurança bradesco - eletrônica). *Bradesco*, . URL https://www.bradescoseguranca.com.br/html/seguranca_corporativa/pf/dispositivos-de-seguranca/token-fisico.shtm. Acesso em: 03 de fev.2019.
- [5] Token santander. *Santander*, . URL <https://www.santander.com.br/portal/wps/script/templates/GCMRequest.do?page=8490>. Acesso em: 03 de fev.2019.
- [6] Verificação em duas etapas. *Google*, . URL <https://www.google.com/landing/2step/>. Acesso em: 03 de fev.2019.
- [7] Two-step verification - overview. *LinkedIn*, . URL <https://www.linkedin.com/help/linkedin/suggested/531/two-step-verification-overview?lang=en>. Acesso em: 03 de fev.2019.
- [8] Como habilitar a verificação em dois passos. *Dropbox*, . URL <https://help.dropbox.com/pt-br/security/enable-two-step-verification>. Acesso em: 03 de fev.2019.
- [9] Verificação em 2 etapas. *Playstation*, . URL <https://www.playstation.com/pt-br/account-security/2-step-verification/>. Acesso em: 03 de fev.2019.
- [10] Multi-factor-authentication. *WatchGuard*, . URL <https://www.watchguard.com/wgrd-products/multi-factor-authentication>. Acesso em: 03 de fev.2019.

- [11] David M'Raihi, Salah Machani, Mingliang Pei, and Johan Rydell. Totp: Time-based one-time password algorithm. *RFC 6238*, maio 2011. URL <https://tools.ietf.org/html/rfc6238>. Acesso em: 03 de fev.2019.
- [12] David M'Raihi, Mihir Bellare, Frank Hoornaert, David Naccache, and Ohad Ranen. Hotp: An hmac-based one-time password algorithm. *RFC 4226*, dezembro 2005. URL <https://tools.ietf.org/html/rfc4226>. Acesso em: 03 de fev.2019.
- [13] Internet engineering task force. *IETF*, . URL <https://www.ietf.org/>. Acesso em: 03 de fev.2019.
- [14] Open authentication. *OATH*, . URL <https://openauthentication.org/>. Acesso em: 03 de fev.2019.
- [15] Hardware authenticators. *Datablink*, . URL <https://www.watchguard.com/wgrd-datablink/financial-services-and-products>. Acesso em: 29 de abr.2019.
- [16] Software authenticators. *Datablink*, . URL <https://www.watchguard.com/wgrd-datablink/financial-services-and-products>. Acesso em: 29 de abr.2019.
- [17] Unix time. *Wikipedia*, fevereiro 2011. URL http://en.wikipedia.org/wiki/Unix_time. Acesso em: 29 de abr.2019.
- [18] He Sun, Kun Sun, Yuewu Wang, and Jiwu Jing. Trustotp: Transforming smartphones into secure one-time password tokens. outubro 2015. doi: <http://dx.doi.org/10.1145/2810103.2813692>.
- [19] Mariano Luis T. Uymatiao and William Emmanuel S. Yu. Time-based otp authentication via secure tunnel (toast): A mobile totp scheme using tls seed exchange and encrypted offline keystore. abril 2014. doi: 10.1109/ICIST.2014.6920371.
- [20] Cullen Linn and Saumya Debray. Obfuscation of executable code to improve resistance to static disassembly. outubro 2003. doi: 10.1145/948109.948149.

AUTORES



Leandro Borges Castilho nasceu em Penápolis, SP, em julho de 1981. Possui os títulos: Engenheiro Eletricista (Inatel, 2005) e Especialista em Gestão Estratégica de Projetos - Metodologia do PMI (FAI, 2008). De 2005 a 2013 trabalhou como Analista de Desenvolvimento de Sistemas no ramo de Software de Gestão – ERP, fazendo implantações, treinando e desenvolvendo em Java de módulos como compras, venda, controle de estoque, produção e folha de pagamento. Desde 2013 a 2017 atuou na área de segurança como Engenheiro de Software, desenvolvendo, em C/C++ e Java, SDK de autenticação baseado em TOTP e soluções de autenticação de segundo fator para protocolos RADIUS, Webservice e login do Windows. Desde 2017 atua como Gerente de Desenvolvimento de Software cuidando da continuidade dos produtos e serviços de autenticação.



Marcelo Vinícius Cysneiros Aragão é graduado em Engenharia de Computação pelo Instituto Nacional de Telecomunicações (Inatel) em 2014 e Mestre em Ciência e Tecnologia da Computação pela Universidade Federal de Itajubá em 2018. Trabalhou de 2011 a 2018 no Inatel Competence Center, mais recentemente como Especialista em Sistemas, onde atuou principalmente como desenvolvedor de soluções de Business Support Systems (BSS) em ambiente de integração contínua. É professor de disciplinas da graduação, como Engenharia de Software e Redes Neurais, e coordenador do curso de pós-graduação em Desenvolvimento de Aplicações para Dispositivos Móveis e Cloud Computing. Possui interesse nas áreas de análise de algoritmos, desenvolvimento de software, inteligência artificial, aprendizado de máquina e ciência de dados.