A rede de confiança em detalhes

Éloïs Librelois
traduzido de https://duniter.fr/wiki/toile-de-confiance/la-toile-de-confiance-en-detail/

O objetivo deste artigo é detalhar o funcionamento dos diferentes parâmetros duma **rede de confiança Duniter**, bem como explicar como os valores desses parâmetros foram **escolhidos no caso da moeda Ğ1**.

Nenhum conhecimento específico é necessário para entender o seguinte, porque a cada vez definirei claramente todos os conceitos discutidos e todos os termos usados. No entanto, iremos longe o suficiente, entraremos nos detalhes do *protocolo Duniter* (apenas no que respeita à rede de confiança, claro), bem como nos aspectos da *teoria dos grafos*, porque me parece essencial entender realmente todo o funcionamento duma rede de confiança Duniter. Não se preocupe, iremos gradualmente, começaremos com uma visão geral e depois detalharemos gradualmente.

Conteúdo

Pré-requisitos

Antes de ler este artigo, é altamente recomendável estudar a licença Ğ1 e ler as seguintes páginas:

- Certificar novos membros
- Perguntas frequentes sobre a rede de confiança

Por que precisamos duma rede de confiança?

Precisamos duma rede de confiança para satisfazer dois objetivos:

1. Assegurar que cada ser humano membro da moeda cocria o mesmo número de Dividendos Universais por intervalo de criação (no caso de Ğ1 este intervalo é `86400` (= 1 dia expresso em segundos), isso é o *parâmetro monetário* `dt`).
2. Identifique os calculadores de blocos para atribuir-lhes uma dificuldade personalizada. Isso é necessário para evitar que o mecanismo de prova de trabalho permita uma centralização do suporte da moeda (a blockchain), *como infelizmente é o caso de muitas criptomoedas não livres*.

Espera, o que é um «*parâmetro monetário*»?

Cada moeda Duniter possui sua própria blockchain dentro da qual são definidos os seus próprios parâmetros monetários (**no bloco zero**), trata-se de um conjunto de «cursores» que devem ser ajustados de acordo com os objetivos visados. No momento em que escrevo estas linhas, o protocolo Duniter (conhecido como DUP) possui **vinte e um parâmetros monetários, dez dos quais relacionados à rede de confiança!** Descreveremos em detalhes os dez parâmetros que dizem respeito à rede de confiança, não discutiremos os outros. Só saibam que no caso da moeda Ğ1, a DU é criada a cada 24h (86400 segundos), mas esse intervalo de tempo, regido pelo parâmetro dt, pode muito bem ser configurado com outros valores para outras moedas.

Não trataremos aqui do segundo objetivo referente à prova de trabalho, pois não é o assunto deste artigo, a única coisa a entender é que a rede de confiança nos permite **identificar** os membros que calculam os blocos, e que essa identificação nos permite impor uma rotação dos membros calculadores; isso não seria possível sem um sistema de identificação. Essa rotação dos membros calculadores é essencial, sem ela um membro muito rico poderia investir em fazendas de computação gigantes para assumir o controle da blockchain e paralisar toda a comunidade!

Então, vamos voltar ao primeiro objetivo, garantir que cada ser humano só tenha apenas uma conta de membro. Na prática, o risco zero não existe, então o nosso objetivo não é conceber uma rede de confiança na qual a fraude seja impossível (isso é impossível de qualquer maneira). Então aqui está uma reformulação mais realista em quatro sub-objetivos:

1. Tornar o ato de certificação pesado o suficiente para exigir que os membros exerçam um mínimo de vigilância.
2. Tornar a fraude difícil o suficiente para ser marginal.
3. Evitar que possíveis ataques Sybil tenham um impacto significativo na moeda (*que os falsos DUs criados têm um peso insignificante em comparação com a massa monetária legítima*).
4. Tornar o crescimento das regiões Sybil lentas o suficiente para que a comunidade tenha tempo de reagir.

Espera, um ataque o quê?

Um ***ataque Sybil*** é o nome dado a um ataque a um sistema de reputação por meio da *criação de identidades falsas*. Uma rede de confiança é um caso especial de um **sistema de reputação**.

Existem muitas estratégias possíveis de cenário de ataque Sybil, bem como diferentes motivos. O nosso objetivo é que a rede de confiança nos proteja contra ataques Sybil que possam comprometer o bom funcionamento da moeda e/ou da rede de computadores que a carrega. O que significa que os micro-ataques Sybil orquestrados por um pequeno grupo, cujo objetivo é apenas um pequeno enriquecimento pessoal, não nos interessam aqui, não cabe à rede de confiança proteger-nos desses micro-ataques, mas à justiça da comunidade interessada, assim como não cabe ao município proteger-nos dum roubo em casa, mas o município garantirá o funcionamento das redes de água, a limpeza das estradas, etc. Da mesma

forma, a rede de confiança Duniter nos garante coletivamente o bom funcionamento da nossa moeda e da rede de computadores que a carrega, e isso já é enorme!

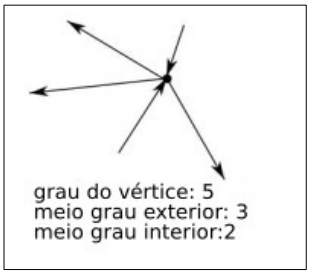
A importância de ser independente de qualquer outro sistema de identificação

Muito regularmente nos é proposto fazer uso de sistemas de identificação centralizados e alimentados por moedas não livres dependentes de um Estado ou outro. Mas então estaríamos dependentes desses sistemas de identificação centralizados para o bom funcionamento da nossa rede de confiança, isso não faz sentido. Além disso, os membros duma moeda livre podem ser de qualquer nacionalidade ou cultura, perderíamos essa universalidade dependendo dum sistema de identificação estatal. Além disso, excluiria as pessoas sem documentos e apátridas. É essencial para nós não dependermos de nenhum estado ou de nenhuma instituição. Só dependemos da rede da internet, e mais, existem outras redes, se a internet moresse, humanos que são membros duma moeda livre poderiam muito bem criar a sua própria rede de informação, isso já é o que algumas associações estão fazendo, criando a sua própria rede, por isso é possível.

Alguns fundamentos da teoria dos grafos

Um pouco de vocabulário

- **grafo:** conjunto de pontos (chamados **vértices** ou **nós**) ligados entre si por setas (chamadas **arcos**).
- **grafo simples:** grafo sem *loop* (arco que liga um vértice a si mesmo) e sem arcos sobrepostos (vários arcos que ligam o mesmo par de vértices na mesma direção).
- **grafo orientado:** grafo cujos arcos têm um sentido, o arco A –> B é portanto diferente do arco B –> A.
- **extremidade inicial** e **final** dum arco: o arco A –> B tem extremidade inicial A e extremidade final B.
- **vértice isolado:** vértice não ligado a nenhum arco.
- **grau dum vértice:** número de arcos que ligam este vértice (em ambas as direções).
- **meio grau exterior** dum vértice A: número de arcos tendo o vértice A como extremidade inicial
- **meio grau interior** dum vértice A: número de arcos tendo o vértice A como extremidade final



- **caminho:** caminho que deve ser percorrido para ir de um vértice A a um vértice B respeitando a direção dos arcos. O número de arcos percorridos é o **comprimento** do caminho.

Definição duma rede de confiança Duniter

As redes de confiança Duniter (uma por moeda) são grafos orientados simples sem vértices isolados. Os vértices são os **membros** e os arcos as **certificações**.

Por que orientado? Certificar é um ato pessoal que só compromete o emissor da certificação, a confiança que ele deposita no receptor não é necessariamente recíproca em todos os casos (muitas vezes é), mas não se pode impor ao receptor a confiança no emissor.

Além disso, todos os vértices são identidades de membros ou já foram membros no passado, os vértices correspondentes aos ex-membros estão em um estado determinado chamado «desativado». Os vértices desativados não podem mais emitir ou receber novas certificações, mas as certificações que emitiram e receberam antes de perder o seu status de membros permanecem ativos para outros membros até a data de expiração, em um esforço para evitar um colapso em cascata da rede de confiança. Se esses ex-

membros não se tornarem membros novamente, as certificações que eles emitiram e receberam acabarão por expirar e os vértices «desativados» em questão se tornarão, portanto, vértices isolados.

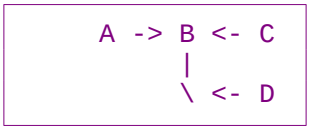
Para finalizar esta história de ex-membro, após um certo atraso que depende dum parâmetro monetário, o vértice desativado é deletado e a identidade associada passa para o estado revogado, ou seja, aquela identidade nunca mais poderá tornar-se membro. O humano que tinha essa identidade, no entanto, permanece livre para se inscrever novamente como membro da rede de confiança sob uma nova identidade.

O que entende por identidade?

Uma identidade é um grupo de **três informações**: uma **chave pública**, um **nome** e um **blockstamp**. Um *blockstamp* é uma referência a um bloco específico duma blockchain. Isso permite datar o momento em que a identidade foi criada e vincular a identidade a uma blockchain específica, portanto a uma moeda específica (porque cada moeda possui a sua própria blockchain).

Uma identidade pode estar em cinco estados diferentes: *pendente, membro, ex-membro, revogado ou excluído*. Voltaremos a cada um desses estados com mais detalhes.

Vamos resumir tudo com um exemplo:



Se, por algum motivo, A perder o seu status de membro, a rede de confiança entrará em colapso e todos os outros membros também perderão o seu status de membro em cascata. Para evitar isso, a certificação A –> B permanecerá válida até a sua data de expiração, dando tempo para que B ser certificado por C ou D, por exemplo.

A ausência dum vértice isolado também implica que quando um novo membro é adicionado à rede de confiança, todas as certificações que permitem que ele se torne membro. É por isso que precisamos dum espaço de armazenamento intermediário que contenha as identidades **pendentes** que aguardam para se tornar membro, bem como as certificações emitidas pelos membros para essas identidades, esse espaço é a famosa «**piscina**» de nós Duniter (que também poderíamos ter chamado de «*caixa de areia*» já que no código de Duniter, *caixa de areia* é o termo usado). Observe que esses «piscinas» também contêm outros tipos de documentos não mencionados aqui.

Definição das regras duma rede de confiança Duniter

As redes de confiança Duniter (uma por moeda) são regidas por **oito regras** apresentadas abaixo. A aplicação dessas regras depende de **onze parâmetros variáveis** de uma moeda para outra. *O valor de dez deles é fixo no bloco zero*, o décimo primeiro parâmetro `msPeriod` é escrito em duro no código, pois foi adicionado após a gravação do bloco zero da Ğ1. Esta parte apresenta apenas as definições das regras, as razões de ser dessas regras são apresentadas no quadro da historicidade do seu aparecimento na parte «*Origem das regras e caso da Ğ1*».

1. Regra de distância e membros referentes

Parâmetros: `StepMax` e `xPercent`

Esses dois parâmetros estão fortemente ligados e juntos definem uma única e mesma regra: **a regra de distância**. Para definir a regra de distância, devemos primeiro definir o que é um membro referente:

- **Membro referente:** um membro A é referente se e somente se os seus dois meio-graus forem maiores ou iguais a `E(N^(1/stepMax))+1` onde N é o número total de membros.

Agora podemos definir a regra de distância:

- **Regra de distância:** um membro A respeita a regra de distância se e somente se para mais de `xPercent` % dos membros referentes R houver um caminho de R para A com comprimento menor ou igual a `stepMax`.

O conceito de membro referente é usado apenas para aplicar a regra de distância, *membros referentes não têm privilégios sobre membros não referentes*. Numa rede de confiança acabada, ou seja, numa rede onde cada membro certificou todos os membros que ele é capaz de certificar legitimamente, todos os membros devem ser referentes, mas devido ao crescimento progressivo da rede por um lado e a substituição geracional, por outro lado, existe em todos os momentos `t` membros que ainda não certificaram todos os membros que podem legitimamente certificar. Esses membros estariam bloqueandos se fossem contados na regra de distância, e a rede de confiança jamais poderia acolher novos membros. (Pode visualizá-lo na página «qualidade da rede» de currency-monit, marcando a opção «*se o conceito de membro referente não existir*»).

Quando se aplica a regra da distância?

Como a verificação da regra de distância é dispendiosa em computações, ela só se aplica ao obter e renovar o status de membro. (Consulte a seção «*Renovação do status de membro*»).

Caso especial: a regra de distância não se aplica ao bloco zero (gravação da rede inicial)

2. Regra do número mínimo de certificações recebidas

Parâmetro: `sigQty`

Esta é a regra mais simples, ela estipula que qualquer membro deve em qualquer momento (ou seja em qualquer bloco) ser o destinatário de pelo menos `sigQty` certificações ativas. Se, mesmo durante um único bloco, o membro A acabar com menos `sigQty` certificações ativas recebidas, ele perderá o status de membro naquele bloco. Ele deve então publicar um *pedido de renovação de sua adesão*.

3. Regra de renovação de associação

Parâmetros: `msValidity`, `msPeriod` e `msWindow`

A obtenção do status de membro não é adquirida para toda a vida, mas por um período de `msValidity` segundos.

Qualquer membro (ou antigo membro não revogado e não excluído definitivamente) pode a qualquer momento fazer um pedido de renovação da sua adesão desde que a sua última renovação seja anterior a mais de `msPeriod` segundos (quando um membro nunca renovou, a data da última renovação corresponde à data de obtenção da qualidade de membro). Quando um pedido de renovação de adesão é emitido, ele é armazenado na «piscina» por no máximo `msWindow` segundos, depois será registrado na blockchain assim que o membro em questão respeitar a regra de distância **e** a regra `sigQty` (e se já as respeitar, assim que um nó que recebeu o pedido de renovação encontra um bloco).

Qualquer membro cuja última renovação data de mais de `msValidity` segundos perde o status de membro no primeiro bloco em que essa duração é atingida. Neste caso, o ex-membro tem novamente um período de `msValidity` segundos para voltar a ser membro por este mesmo procedimento de renovação. Após este período, ou seja, `2 × msValidity` após a última renovação, o ex-membro é excluído definitivamente e não poderá voltar a ser membro com esta conta. Se ele deseja se tornar um membro novamente, terá que criar uma nova conta do zero.

4. Regra de validade da certificação

Parâmetro: `sigValidity`

Qualquer certificação registrada na blockchain expira `sigValidity` segundos após sua emissão.

> A emissão e a gravação de uma certificação ocorrem em momentos diferentes. Quando o membro A emite uma certificação em um tempo t_1, ele primeiro é armazenado na piscina neste momento t_1, depois será gravado na blockchain em um tempo t_2, assim que as regras da rede permitirem, pode haver vários semanas de diferença entre t_1 e t_2!

5. Regra do estoque limitado de certificações ativas

Parâmetro: `sigStock`

Por certificação ativa entendemos a certificação *registrada na blockchain e que ainda não expirou*.

Em qualquer bloco e para qualquer membro, todas as certificações emitidas por este membro e ativas devem ser menores ou iguais a `sigStock`. Quando esta quota for atingida, o membro em questão terá de esperar que uma certificação ativa que ele emitiu expire para poder emitir outra.

6. Regra do intervalo de gravação entre duas certificações

Parâmetro: `sigPeriod`

Quando uma certificação emitida por um membro A é gravada na blockchain, nenhuma outra certificação emitida por A pode ser escrita na blockchain por `sigPeriod` segundos.

7. Regra da janela de gravação de certificação

Parâmetro: `sigWindow`

Quando uma certificação é emitida por um membro A, ela permanecerá armazenada na «piscina» por no máximo `sigWindow` segundos. Se a certificação em questão ainda não tiver sido gravada na blockchain após esse período, ela será totalmente excluída.

8. Regra da janela de gravação de identidade

Parâmetro: `idtyWindow`

Quando uma identidade é emitida, ela permanecerá armazenada na «piscina» por no máximo idtyWindow segundos. Se a identidade em questão ainda não tiver sido gravada na blockchain após esse período, ela será pura e simplesmente excluída.

O caso especial da rede inicial no bloco zero

A aplicação das regras apresentadas apenas torna possível a expansão de uma rede a partir de uma rede pré-existente, portanto, há um caso especial em que algumas regras não se aplicam: escrever bloco zero.

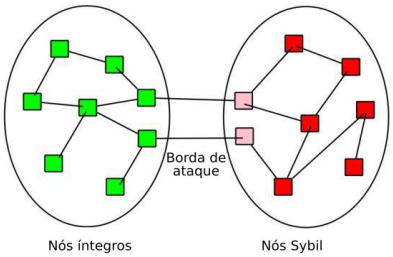
Somente as regras **2.** e **5.** se aplicam ao gravar o bloco zero.

Na prática, é o humano que gera o bloco zero que escolhe manualmente quais identidades serão gravadas no bloco zero, mas todas as identidades e certificações gravadas no bloco zero devem seguir as regras 2. e 5. e adicionalmente o bloco zero deve ser assinado com a chave privada duma das identidades gravadas. Assim que um bloco zero correto for gerado, qualquer identidade registrada no bloco zero pode propor o bloco seguinte, de fato, o autor do bloco zero não tem mais controle.

Origem das regras e casos do Ğ1

1. Distância e limite de tamanho

O objetivo da regra de distância é limitar o tamanho máximo duma região Sybil, bem como o tamanho máximo da comunidade monetária. `xPercent` permite evitar uma minoria de bloqueio (membros pouco ativos).



As regiões Sybil estão isoladas do restante do grafo, pois as contas Sybil receberão apenas certificações de outras contas Sybil ou dos autores do ataque. Assim, qualquer caminho mais curto entre um membro legítimo e uma conta Sybil passa necessariamente por um autor do ataque. O limite da profundidade da região Sybil, portanto, depende da distância máxima entre os autores do ataque e os `xPercent`% membros referentes mais próximos, essa distância característica é denominada `stepAttackers`. O tamanho máximo de uma região Sybil criada por `sigQty` membros mal-intencionados depende da alavanca `L=sigQty/sigStock`:

```
taille max région Sybil = (sigStock-sigQty)*(1-L^(stepMax-stepAttackers))/(1-L)
```

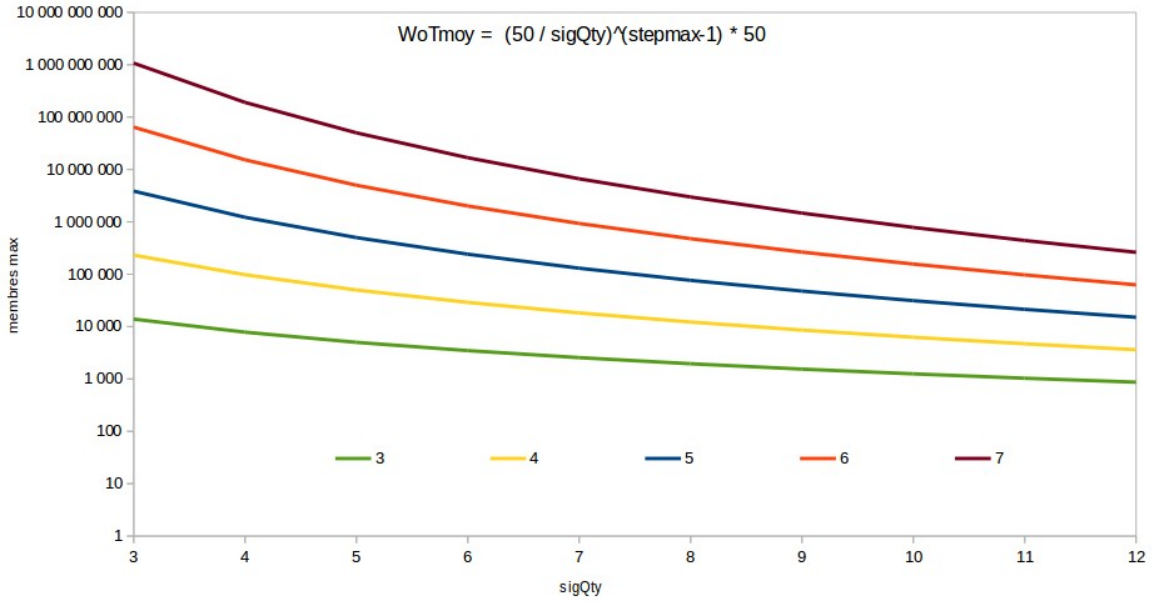
E, da mesma forma, o tamanho máximo teórico da rede de confiança é:

```
WoTmax = (sigStock)*L^(stepMax-1)
```

Só que, na realidade, a maioria dos membros não consumirá todo o seu estoque de certificações. Muitos estudos sociológicos mostram que o ser humano conhece, em médio, cinquenta pessoas, então vamos substituir sigStock por 50:

```
WoTmoy = (50)*(sigQty/50)^(stepMax-1)
```

A nossa rede de confiança, portanto, pode ser dimensionada com apenas dois parâmetros, o nosso objetivo com a Ğ1 é criar uma zona econômica livre *da ordem de um milhão de usuários*, então vamos ver quais combinações do par `(sigQty;stepMax)` nos permitem esperar o milhão:



O tamanho máximo de uma região Sybil cresce *linearmente* em relação a `sigQty`, mas *exponencialmente* em relação a `stepMax`, portanto, para obter a rede mais forte possível, precisamos minimizar o valor de `stepMax`. O gráfico acima nos mostra que o menor valor `stepMax` para chegar a um milhão de membros é **cinco**. Para o valor de `sigQty`, temos a escolha entre *quatro* para chegar a 1,2 milhão ou *cinco* para chegar a meio milhão. Sabendo que a fórmula `WoTmoy` nos dá uma ordem de grandeza e que considera todas as contas como referentes (o que não é o caso na realidade), o tamanho máximo real que poderemos atingir é certamente superior, meio milhão é portanto suficiente, especialmente como mais `sigQty` é fraco, mais fácil é lançar um ataque Sybil (é mais fácil ser quatro cúmplices do que cinco). Como medida de segurança, escolhemos **cinco**, o que já nos dá:

```
stepMax = 5, sigQty = 5, sigStock ≥ 50
```

o tamanho máximo de uma região Sybil é, portanto:

```
(sigStock-sigQty)*(1-(sigStock/5)^(5-stepAttackers))/(1-(sigStock/5))
```

com `sigStock = 50` isso faz uma região Sybil de: `45*(1-10^(5-stepAttackers))/(-9)`

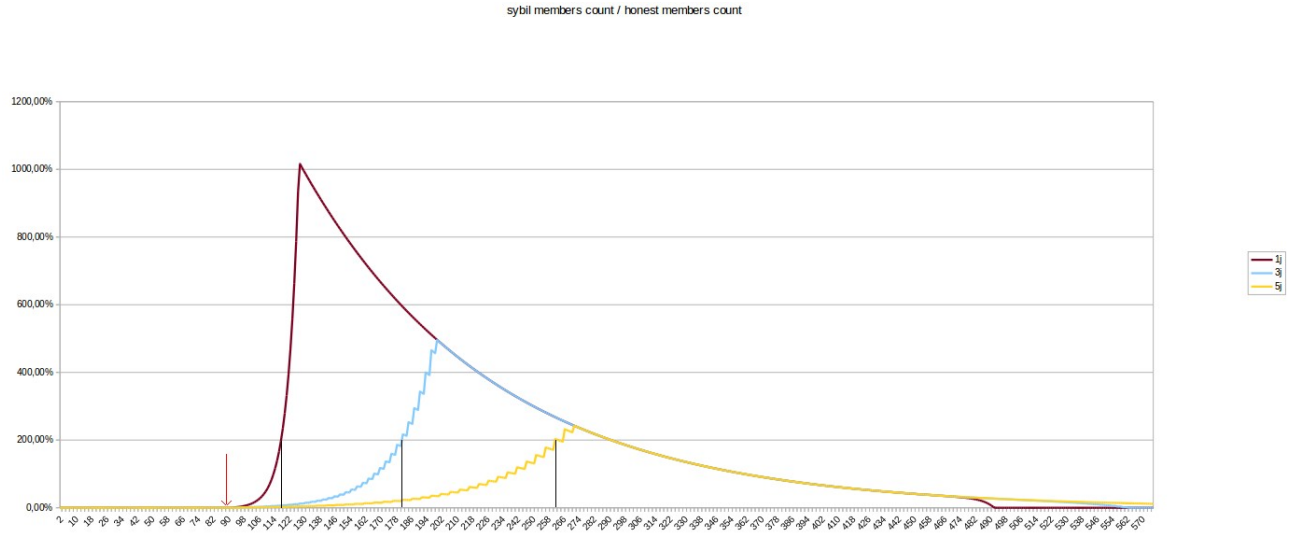
Uma boa maneira de nos proteger é maximizar `stepAttackers`. É por isso que nos asseguramos de que na rede inicial Ğ1 muitos membros de referência já estejam a uma distância de *quatro* um do outro.

Outra maneira de nos proteger de um ataque Sybil, se fosse lento o suficiente para termos tempo de detectá-lo, os membros de referência poderiam voluntariamente estender a rede para impedir que a zona Sybil se expandisse até que os autores do ataque fossem privados da sua condição de membro por não renovação das certificações legítimas das quais eles se beneficiaram para se tornar um membro. O problema é que um ataque Sybil pode ser propagado por contas de bot que irão certificar de forma muito rápida e otimizada, então como poderíamos desacelerá-los à força? *Ao impor um período mínimo entre duas certificações*!

2. Uma questão de tempo

Acabamos de ver que, para desacelerar a implantação de um ataque Sybil para ter tempo de detectá-lo e tomar medidas, poderíamos impor *um atraso mínimo entre duas gravações de certificação* pela mesma conta. É por isso que criamos o parâmetro `sigPeriod`.

Aqui está um gráfico da evolução do tamanho de uma região Sybil em função de `sigPeriod`:



Vemos que o parâmetro `sigPeriod` tem uma influência muito forte na velocidade de propagação duma região Sybil, então precisamos escolher um `sigPeriod` alto o suficiente para que as áreas legítimas da rede possam crescer pelo menos tão rápido quanto as regiões Sybil.

Além disso, uma `sigPeriod` alta torna o ato de certificação mais caro, e quanto mais caro esse ato, mais ele incentiva o usuário a certificar com parcimônia. Portanto, há muitas vantagens em escolher um `sigPeriod` alto e nenhuma desvantagem técnica, então optamos por **5 dias**.

Também poderíamos ter escolhido 7 dias (=1 semana) por uma questão de simplicidade, mas outra ideia subjaz à escolha de 5 dias em vez de 7: os ritmos das nossas sociedades. A certificação exige tempo para garantir a conformidade com a licença Ğ1, e a maioria dos humanos tem mais tempo nos fins de semana, então a ideia era permitir que um usuário que emitiria 1 certificação por fim de semana não fosse limitado por `sigPeriod`, portanto, estabelecemos a duração de um semana de trabalho (5 dias), em vez de uma semana completa (7 dias).

3. Uma confiança eterna? (`sigValidity`, `msValidity`)

Se todas as certificações permanecessem válidas *ad vitam æternam*, isso seria problemático por pelo menos duas razões: por um lado, é importante que os membros que falecem parem de criar DUs; por outro lado, é importante que contas falsas detectadas pela comunidade não permaneçam como membros indefinidamente.

Para isso, é necessário que as certificações tenham uma vida útil limitada e que, portanto, seja necessário renovar regularmente a confiança nos seus pares. Por outro lado, também não é necessário que os membros gastem o seu tempo renovando as suas certificações em vez de participar de trocas econômicas. Além disso, uma certificação com vida útil muito curta tornaria o ato de certificação leve demais. O ato de certificação deve permanecer consistente para ser levado a sério. Por fim, também queremos que a vida útil duma certificação seja fácil de lembrar, para que o maior número possível de pessoas se lembre dela.

Tendo em conta todos estes critérios, hesitamos entre um, dois ou três anos. Historicamente, definimos primeiro os valores de `sigPeriod` e `sigStock`, o que fez com que levasse *pelo menos 495 dias* para esgotar todo o seu estoque de certificações, portanto um ano não é possível. Três anos nos pareceram muito longos, então escolhemos **dois anos**.

Mas considerando que um morto continuará criando DUs que trazerão vantagens para ninguém por um período de até dois anos, isso nos pareceu demais. É por isso que optamos por uma vida útil menor para a renovação da adesão. Então escolhemos **um ano**. Mas também poderíamos ter escolhido seis meses. O valor de `msValidity` é bastante secundário e pode ser facilmente alterado no futuro, se a comunidade desejar.

4. Evitar que as piscinas se transformem em depósitos sedimentares (`idtyWindow`, `sigWindow`, `msWindow`)

As piscinas precisam ser limpadas regularmente para evitar que atinjam tamanhos astronômicos e para garantir que mesmo máquinas pequenas e com pouca potência possam executar um nó Duniter. Para fazer isso, as identidades e certificações pendentes devem permanecer na piscina *pelo menor tempo possível*; no entanto, devem permanecer lá por tempo suficiente para ter uma chance razoável de serem registradas na blockchain. Para a Ğ1, estimamos que dois meses era o mínimo para que todos os potenciais certificadores duma nova identidade tivessem tempo para sincronizar. Também queríamos que essa duração correspondesse a um pequeno número de unidades usuais (*ou seja: dias, meses, anos em vez de segundos*), de forma a ser facilmente integrada e memorizada pelo maior número de pessoas possível. Tipicamente, escolher sete semanas teria sido mais complicado de integrar e lembrar. Queríamos que as durações conhecidas permanecessem tão fáceis de integrar e memorizar quanto possível. Um mês nos pareceu muito curto, então o valor de **dois meses** foi imposto. E para simplificar, aplicamos esse mesmo valor de dois meses a todos os três parâmetros `idtyWindow`, `sigWindow` e `msWindow`.

5. Limitar membros que conhecem muitas pessoas (`sigStock`)

Muitos estudos sociológicos mostram que um ser humano conhece em média *cinquenta* pessoas. Claro, isso é uma média. Alguns humanos conhecem muito mais, outros muito menos. Aqui, novamente, estabelecemos o critério de «*número fácil de lembrar*». Embora o impacto de `sigStock` no tamanho máximo das regiões Sybil seja secundário, ainda é aconselhável não escolher `sigStock` muito grande! 150 parecia muito alto para nós, então escolhemos **100**.

6. Protejar-se contra a minoria de bloqueio (`xPercent`)

É muito fácil tornar-se membro referente, então um possível ataque poderia consistir em criar uma região Sybil de membros referentes. Essa região Sybil cresceria muito mais lentamente, mas poderia bloquear o restante da rede legítima jogando com a regra da distância. Esta é a razão pela qual a regra de distância não deve ser baseada em 100% dos membros indicados. Então decidimos criar um parâmetro `xPercent` que defina qual proporção de membros referentes deve estar *dentro de 5 passos de cada um*.

Essa proporção deve ser baixa o suficiente para evitar uma minoria de bloqueio (de membros referentes Sybil que estariam muito longe dos membros legítimos). Mas, paradoxalmente, essa proporção também deve ser alta o suficiente para que a regra de distância limite adequadamente o tamanho máximo das regiões Sybil. O parâmetro `xPercent` foi um dos mais difíceis de definir, por isso nos permitimos ajustar o seu valor ao longo do experimento Ğ1 se acharmos necessário.

Fomos inspirados pelo **princípio 80-20**: se pelo menos 20% dos membros cuidarem de densificar as áreas legítimas da rede de confiança, então 80% dos referentes devem estar a 5 passos de cada membro. 80% é, portanto, um valor máximo para `xPercent`, acima deste valor, a regra de distância pode ser muito restritiva em usos legítimos. Por questões de segurança, optamos pelo valor máximo possível.

7. Adicionando `msPeriod` como proteção anti-spam

Este parâmetro um pouco separado, *adicionado após o bloco zero*, não é usado para regular a rede de confiança, mas para proteger a rede informática que veicula a moeda contra ataques do tipo «spam». De fato, membros mal-intencionados que desejam prejudicar o bom funcionamento da moeda podem solicitar a sua renovação a cada bloco (atualmente, a cada **cinco minutos**) ou, pior ainda, enviar centenas de solicitações de renovação por minuto para sobrecarregar os nós Duniter. Se não houver limitação, os nós Duniter deveriam processar todas os pedidos de renovação, mesmo que a adesão tenha sido renovada há cinco minutos! Para proteger contra essa possibilidade de ataque, adicionamos o parâmetro msPeriod. Para simplificar, optamos por atribuir a ela o mesmo valor dos parâmetros `idtyWindow`, `sigWindow` e `msWindow`.

Os onze parâmetros variáveis da Ğ1

```
StepMax = 5 passos
sigQty = 5 certificações
sigStock = 100 certificações
sigPeriod = 5 dias
sigValidity = 2 anos
msValidity = 1 ano
idtyWindow = 2 meses
sigWindow = 2 meses
msWindow = 2 meses
xPercent = 80%

msPeriod = 2 meses
```