```php
<?php

declare(strict_types=1);

namespace Plugin\fp_altcha_spamschutz\src\Service;

use AltchaOrg\Altcha\V1\Altcha;
use AltchaOrg\Altcha\V1\ChallengeOptions;
use AltchaOrg\Altcha\V1\Hasher\Algorithm;
use JTL\Plugin\PluginInterface;

/**
 * Kapselt die ALTCHA V1 Proof-of-Work Bibliothek (MIT-Lizenz,
https://github.com/altcha-org/altcha-lib-php).
 *
 * Warum V1 statt der aktuellen ALTCHA-Widget-Version (v3)?
 * Das offizielle JS-Widget v3 nutzt ein neueres, komplexeres Protokoll
(PBKDF2/Argon2id/Scrypt mit
 * Key-Prefix-Matching), fuer das es keine einfache, robust dokumentierte
"Challenge direkt einbetten"-
 * Variante ohne zusaetzlichen Netzwerk-Endpunkt gibt. Das klassische V1-
Protokoll (SHA-256 Hashcash:
 * Client sucht per Brute-Force eine Zahl n, fuer die SHA256(salt+n) ==
challenge gilt) ist dagegen
 * vollstaendig dokumentiert, einfach zu pruefen und wird hier zusammen mit
einem kleinen eigenen
 * JS-Loeser (assets/fp-altcha.js) eingesetzt. Serverseitig kommt weiterhin
der offizielle,
 * unveraenderte ALTCHA-Code zum Einsatz (siehe
src/Vendor/AltchaOrg/Altcha/V1).
 */
class AltchaService
{
    private PluginInterface $plugin;
    private Altcha $altcha;

    public function __construct(PluginInterface $plugin)
    {
        $this->plugin = $plugin;
        $this->altcha = new Altcha($this->getHmacSecret());
    }

    public function isConfigured(): bool
    {
        return $this->getHmacSecret() !== '';
    }

    public function isEnabledForRegistration(): bool
    {
        return $this->getConfigValue('fp_altcha_protect_register', 'on') ===
```

```
'on';
    }

    public function isEnabledForNewsletter(): bool
    {
        return $this->getConfigValue('fp_altcha_protect_newsletter', 'on')
=== 'on';
    }

    public function isDebug(): bool
    {
        return $this->getConfigValue('fp_altcha_debug', '') === 'on';
    }

    /**
     * Erzeugt eine neue Pruefung und liefert sie als Array, das 1:1 als
JSON in die Seite
     * eingebettet werden kann (siehe TemplateHandler).
     *
     * @return array<string, string|int>
     */
    public function createChallengeArray(): array
    {
        $maxNumber = (int) $this->getConfigValue('fp_altcha_max_number',
'150000');
        if ($maxNumber < 1000) {
            $maxNumber = 150000;
        }

        $expirySeconds = (int)
$this->getConfigValue('fp_altcha_expiry_seconds', '600');
        if ($expirySeconds < 30) {
            $expirySeconds = 600;
        }

        $expires = new \DateTimeImmutable('+' . $expirySeconds . '
seconds');

        $challenge = $this->altcha->createChallenge(new ChallengeOptions(
            algorithm: Algorithm::SHA256,
            maxNumber: $maxNumber,
            expires: $expires,
        ));

        return [
            'algorithm' => $challenge->algorithm,
            'challenge' => $challenge->challenge,
            'maxnumber' => $challenge->maxNumber,
            'salt' => $challenge->salt,
            'signature' => $challenge->signature,
```

```
        ];
    }

    /**
     * Prueft das per POST["altcha"] gesendete, base64-kodierte Loesungs-
Payload.
     */
    public function verifyPost(): bool
    {
        $field = $_POST['altcha'] ?? null;

        if (!\is_string($field) || $field === '') {
            $this->log('kein altcha Feld im POST gefunden');

            return false;
        }

        if (!$this->isConfigured()) {
            // Kein HMAC-Secret hinterlegt -> Plugin ist nicht korrekt
eingerichtet.
            // Sicherheitshalber ablehnen statt durchzulassen.
            $this->log('kein HMAC-Secret konfiguriert, Pruefung wird
abgelehnt');

            return false;
        }

        $verified = $this->altcha->verifySolution($field, true);
        $this->log('Pruefungsergebnis: ' . ($verified ? 'erfolgreich' :
'fehlgeschlagen'));

        return $verified;
    }

    private function getHmacSecret(): string
    {
        return $this->getConfigValue('fp_altcha_hmac_secret', '');
    }

    private function getConfigValue(string $name, string $default): string
    {
        $value = $this->plugin->getConfig()->getValue($name);

        if (!\is_string($value) || $value === '') {
            return $default;
        }

        return $value;
    }

    private function log(string $message): void
```

```
    {
        if (!$this->isDebug()) {
            return;
        }

        error_log('[fp_altcha_spamschutz] ' . $message);
    }
}
```